МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Інформаційна система обліку і відвантаження рідких вантажів»
за спеціальністю 122 «Комп'ютерні науки та інформаційні технології»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ-51-6 Козак Олександр Володимирович

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «__» _____2019 р.

Науковий керівник _____ к.т.н., доц.,Шендрик В.В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д.М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Напрям підготовки – 6.050101 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__»_____ 2019 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Козак Олександр Володимирович

1 Тема роботи *Інформаційна система обліку і відвантаження рідких вантажів*

керівник роботи *Шендрик Віра Вікторівна, к.т.н., доцент*

затверджені наказом по університету від «__»_____2019 р. № 0834-III

2 Строк подання студентом роботи «__»_____2019 р.

3 Вхідні дані до роботи технічне завдання, вимоги до оформлення дипломної роботи, методичні вказівки

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, постановка задачі, проектування системи, розробка кінцевого продукту

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Розробка бази даних.	3 дні	
2	Розробка функціональних блоків web додатку	8 днів	
3	Розробка API для роботи з віддаленими пристроями	8 днів	
4	Розробка інтерфейсу	4 дні	
5	Підбір функціональних блоків для пристрою обліку і відвантаження	2 дні	
6	Підключення необхідних бібліотек	2 дні	
7	Розробка програми для мікроконтролера	7 днів	
8	Тестування	4 дні	

Студент _____ Козак О.В.
(підпис)

Керівник роботи _____ к.т.н., доц. Шендрик В. В.
(підпис)

РЕФЕРАТ

Записка: 90 стор., 54 рис., 4 додатка, 17 джерел інформації.

Мета роботи — створення інформаційної системи обліку і відвантаження рідких вантажів.

Методи реалізації — web додаток, працюючий в парі з віддаленим пристроєм обліку.

Результати — розроблено систему обліку і відвантаження рідких вантажів, перевірено її працездатність.

У першому розділі було розглянуто основні принципи побудови автоматизованих інформаційних систем подібного призначення, розглянуто існуючі системи автоматизованого відвантаження рідини, виявлено їх недоліки.

У другому розділі було сформовано мету і задачі проекту, обрано засоби реалізації, розглянуто можливості та характеристик обраних засобів реалізації.

У третьому розділі було здійснено струкрутно-функціональне моделювання процесу відвантаження рідини, було спроектовано базу даних для функціонування системи та визначено варіанти використання системи.

Четвертий розділ описує налаштування середовища розробки, процес підключення необхідних модулів. Описано результат розробки та наведено функціональні можливості розробленої інформаційної системи

Ключові слова: ВІДВАНТАЖЕННЯ РІДИНИ, БЕЗОПЕРАТОРНА СИСТЕМА, ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗОВАНИЙ ОБЛІК, ВЕНДІНГ, WEB ДОДАТОК, СИСТЕМА ОБМЕЖЕНОГО ДОСТУПУ

ЗМІСТ

ВСТУП 6
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7
1.1 Загальний аналіз вендінгових апаратів 7
1.2 Аналіз проблематики предметної області 8
1.3 Особливості роботи і побудови вендінгових апаратів 9
2 ПОСТАНОВКА ЗАДАЧІ 12
2.1 Мета та задачі 12
2.2 Вибір методів реалізації інформаційної системи 13
3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 15
3.1 Структурно-функціональне моделювання процесу відвантаження рідини 15
3.2. Моделювання бази даних інформаційної системи 21
3.3 Моделювання варіантів використання інформаційної системи 34
4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ 37
4.1 Підготовка середовища та розробка компонентів 37
4.2. Результат реалізації інформаційної системи 44
ВИСНОВОК 57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 58
ДОДАТОК А 60
ДОДАТОК Б 71
ДОДАТОК В 79
ДОДАТОК Г 83

ВСТУП

У сучасному світі майже усі процеси у повсякденному житті, сфері послуг та у виробництві підлягають оцифровуванню. Люди використовують месседжери замість паперових листів, таксі або піцу можна замовити через інтернет, а підприємства використовують програми обліку замість ведення паперової звітності. Заправні станції та склади рідких вантажів також потребують впровадження цифрових технологій.

Зараз все ширше використовуються безоператорні системи, існують банки які не мають відділень, а функціонують через мережу інтернет.

Вендінгові апарати без участі касира або оператора можуть виконувати продаж товарів, приготування напоїв, надання деяких послуг. Наприклад, майже усі станції зарядки для електромобілів функціонують автономно, існують кавові автомати, автомати для продажу питної води. [9]

У більш розвинутих країнах вендінгові апарати вже давно перейняли на себе велику частину сфери продажу та послуг і це логічно бо у порівнянні з роботою людини вони мають ряд переваг:

- Вендінгові апарати можуть працювати 24 години на добу, 7 днів на тиждень, лише з перервати на технічне обслуговування;
- Заміняють людину, а отже не потрібно платити заробітну плату;
- Усі дії автомата чітко прописані, виключається людський фактор;
- Якщо підключити автомат до мережі інтернет, то ним можна буде керувати з будь якого місця у світі;
- Автомат генерує зручну електронну звітність;

Інформаційна система обліку і відвантаження рідких вантажів це сукупність вендінгових апаратів і віддаленого серверу. Таку систему можна використовувати як у промисловості для обліку вантажу до і після його перевезення так і для роздрібної торгівлі харчовими і нехарчовими рідинами.

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальний аналіз вендінгових апаратів

В Україні і світі стрімко розвиваються автоматизовані системи обслуговування – вендінгові апарати, найшвидше розвиваються автомати у сфері харчової промисловості, тому купити цілий кавовий автомат або запчастини для нього зараз не є проблемою. [8]

Однак якщо брати специфіку відвантаження рідких вантажів, то вендінгів для заправки автомобілів так легко не знайти. Так на ринку є апарати які можуть працювати автономно, надавати доступ за особистими ключами і мати підключення до локальної мережі.

Прикладом такої колонки є модель Piusi Cube 70MC, від італійського виробника. Але незважаючи на високу ціну в цій моделі також є ряд недоліків.

Такі апарати призначені для використання на підприємстві, адміністрування виконується лише з певного комп’ютера всередині локальної мережі підприємства, також вони мають обмежену кількість користувачів, обумовлену розміром пам’яті у платі керування колонкою, зазвичай до 50 користувачів. Ще один недолік, що такі колонки не призначені для суміжної роботи, кожна колонка адмініструється окремо.

Інші моделі на ринку, від інших виробників також не дозволяють виконувати віддалене адміністрування. А якщо розглядати систему як для роздрібної торгівлі то апарати які були б обладнані купюроприймачами або POS терміналами у вільному продажу взагалі немає.

Тому розробка систем роздрібної торгівлі точно залишається актуальною, оскільки може користуватися попитом у малого і середнього бізнесу.

1.2 Аналіз проблематики предметної області

У сфері перевезення рідких вантажів також є ряд невирішених проблем, досить гостро постала проблема крадіжки частини вантажу під час його транспортування, найчастіше так вчиняють з продуктами нафтопереробного виробництва або харчовими продуктами.

Часто це роблять самі водії або особи з якими водії в домовленостях, не допомагає навіть зважування авто перед виїздом у рейс, та опломбування вхідних та вихідних каналів з автоцистерни.

Пломби підроблюють а різницю у вазі компенсують баластом. Найчастіше так стається тому що кінцеві пункти розвантаження не обладнані пристроями обліку і неправомірні дії залишаються безкарними.

Автоматизовані системи покликані вирішити цю проблему, під час завантаження буде відомий точний об’єм вантажу, дані про його об’єм будуть зберігатися на сервері, а під час розвантаження кількість вантажу буде знову виміряно і при розбіжності значень махінації буде виявлено.

Ще однією найчастішою проблемою є велика кількість технічного персоналу, впровадження автоматизованих систем – вендінгів, дозволить позбутися технічного персоналу повність або частково. Такий крок дозволить у деякій мірі зекономити кошти, що дозволяє знизити ціну на товари та послуги, що в свою чергу підвищує конкурентну спроможність товарів і послуг які надає компанія.

Якщо розглядати звітність, то традиційна паперова або електронна звітність формується за певний час, це може бути доба, або тиждень, або більший проміжок часу, а оскільки дані збираються певний час, а потім їх ще треба обробити, дані втрачають свою актуальність і тому саме автоматична електронна звітність має дуже великі переваги.

Чітка електронна звітність у режимі реального часу дозволяє якісно планувати логістику, проводити аналітичні і статистичні розрахунки.

1.3 Особливості роботи і побудови вендінгових апаратів

У загальному випадку вендінгові апарати можуть працювати або цілком автономно, або автономно з можливістю віддаленого адміністрування. У першому випадку виникає ряд труднощів і недоліків. У разі виникнення несправностей апарат буде простоювати до моменту його чергового огляду, що може складати досить значний проміжок часу, або до того моменту коли про несправність повідомить один з користувачів, але і в першому і другому випадку апарат буде простоювати деякий час, що викликає збитки зі сторони власника апарату. Також проблемою буде те що дані звітності треба буде знімати вручну під час кожного планового огляду. Те саме стосується і зміни налаштування аппарату, їх треба буде виконувати біля кожного апарату в ручному режими. А якщо апаратів багато то для того щоб їх усі переналаштувати піде багато часу.

Якщо ж вендінговий апарат буде мати віддалений доступ, або через дротове з'єднання, або з використанням модему усі проблеми легко вирішуються. Якщо виникне несправність і апарат вчасно зробить запит на сервер отже очевидно в ньому виникла несправність, і адміністратор системи направить людину для ремонту. Звітність одразу буде відправлятися на сервер, а налаштування можна буде оновлювати лише змінивши значення в адміністративній панелі. [10]

Отже система відвантаження рідких вантажів повинна складатися з таких основних блоків:

- WEB додаток на віддаленому сервері, який обробляє запити пристроїв обліку, та на якому зберігаються журнали звітності і налаштування;
- Плата керування вендінгом яка реалізує його віддалену роботу, контролює відправлення звітності, перевіряє картки доступу;
- Модем або мережева картка для під'єднання до мережі інтернет;
- Додаткові пристрої для безпосереднього перекачування рідини (насос, клапани, імпульсний витратомір).

Розглянемо кожний пункт детально.

WEB додаток можна розробити з нуля, а можна використати один з готових фреймфорків. У першому випадку є можливість розробити систему максимально налаштовану під наші потреби, але на розробку систем з нуля піде більше часу. Фреймворки мають вже готові базові системи, містять в собі різні шаблони для більш швидкої розробки, але в противагу нам складно буде підлаштувати все під себе на всі 100%, у деяких випадках треба буде шукати обхідні шляхи для вирішення задачі, що в свою чергу призведе до втрати продуктивності.

Зважаючи на потенційно велику кількість користувачів і пристроїв обліку краще щоб система максимально відповідала нашим потребам, тому раціональним буде розробити повністю свою систему.

Для зберігання даних у нашому випадку цілком підійде реляційна база даних MySQL.

Плата керування у подальшому повинна бути виготовлена серійно, але перед виготовлення серійної плати, треба зробити прототип плати та протестувати на ньому роботу програмного забезпечення для мікроконтролера. Для побудови прототипу на сьогоднішній день існує багато плат для розробки і відлагодження, найвідомішими з них є плати STM на базі процесора ARM, та плати Arduino на базі процесорів AVR.

Враховуючи велику спільноту розробників, велику кількість проектів, і готових бібліотек доцільно використовувати плати Arduino, [10] хоча у порівнянні з платами STM, Arduino програють у швидкодії.

Модем можна брати майже будь який, головне щоб він підтримував GPRS з'єднання, підтримував IP протокол, мав послідовний порт RS232 та ним можна було б керувати за допомогою AT команд.

Вимоги до додаткового обладнання (насос, клапани, імпульсний витратомір), будуть визначатися рідиною яку треба відвантажувати та умовами навколишнього середовища в якому планується експлуатація пристрою обліку.

Оскільки керування блоками додаткового обладнання буде виконуватися за допомогою реле, то безпосередньо моделі додаткового обладнання великої ролі не відіграюсь.

Важливим є лише потужність споживачів які будуть підключені, головне щоб вона не виходила за межі номінальних значень реле.

З програмної точки зору записи про виконання відвантаження, повинні записуватися на сервер автоматично, після завершення операції відвантаження.

На рівні бази даних система повинна буде маніпулювати кортежами даних, обов’язкові дані що повинні міститися в кортежі:

- Унікальний ідентифікатор запису;
- Ідентифікатор пристрою обліку;
- Обліковий запис за допомогою якого отримали доступ до системи;
- Ідентифікатор типу рідини;
- Об’єм відвантаженої рідини;
- Дата та час проведення операції;

Маючи багато записів можна виконувати розрахунки про залишок рідкого вантажу кожному зі складів. Розраховувати потреби товару у тому чи іншому регіоні, планувати найвигідніші маршрути поставок.

Отже враховуючи, що на сьогоднішній день немає пристроїв обліку які б можна було об’єднувати в зв’язану систему яку можливо віддалено адмініструвати, а у сучасному світі йде глобальна тенденція до автоматизації та заміщенню людської праці машинами, власна розробка інформаційної системи обліку і відвантаження рідких вантажів є доцільною та необхідною.

ПОСТАНОВКА ЗАДАЧІ

2.1 Мета та задачі

Інформаційна система повинна забезпечувати автоматизацію роботи пункту відвантаження.

Функціональні вимогу до інформаційної системи, забезпечити:

- додавання записів в журнал обліку пристроями обліку;
- редагування та видалення записів у журналі при виникненні помилок;
- пошук записів в журналі по певним користувачам та іншим критеріям;
- додавання, редагування та видалення користувачів в системі, призначення квот і лімітів;
- отримання звіту по кількості відвантаженого товару по кожній точці відвантаження рідкого вантажу;
- забезпечити функціонал для резервного копіювання даних.

Нефункціональні вимоги до інформаційної системи:

- Коректно працювати при великому навантаженні;
- Забезпечувати цілісність даних;
- Мати зручний, інтуїтивно зрозумілий інтерфейс.

Метою дипломного проекту є розробка і впровадження в роботу інформаційної системи для обліку і відвантаження рідких вантажів.

Інформаційна система повинна вирішити такі задачі:

- заборона несанкціонованого доступу до системи відвантаження;
- створення електронної звітності;
- автоматизація процесу відвантаження;
- скорочення витрат на обслуговування.

В результаті обговорення цілей і задач було сформовано та затверджено технічне завдання, яке розміщене у додатку А.

2.2 Вибір методів реалізації інформаційної системи

Для реалізації програмної частини існує велика кількість мов програмування, СУБД, веб-серверів та фреймворків на яких можна створити веб-додаток будь-якої складності. Важливо грамотно обрати засоби для реалізації ІС. Враховуючи поставлені вище завдання та попередній досвід найбільш оптимальним засобом для розробки такого продукту є застосування мови розмітки документів у Всесвітній павутині – HTML, формальної мови опису зовнішнього вигляду документа, написаного з використанням мови розмітки – CSS, прототипно-орієнтованої сценарного типу мови програмування JavaScript, скриптової мови програмування загального призначення PHP, веб-сервера APACHE та баз даних MySQL.

HTML – мова розмітки гіпертекстових документів. HTML-документи лежать в основі Веб, і відображаються із допомогою веб-браузерів. Разом із видимою інформацією, HTML-документи містять додаткові метадані, такі як, мова тексту, автор документа, стислий підсумок. [6]

Каскадні таблиці стилів (CSS) - це потужний інструмент для формування зовнішнього вигляду веб-сайту, впливає на уявлення документа або набору документів.

PHP – мова програмування, створена для генерування HTML-сторінок на веб-сервері і роботі з базами даних. [5]

В даний час, майже усіма хостинг- провайдерами. В області програмування для веб, PHP — одна з популярних скриптових мов завдяки своїй простоті, швидкості виконання, багатій функціональності і великій кількості супровідної документації.

PHP не передається на сторону клієнта як JavaScript, а виконується веб-сервером, результатом виконання є HTML-код, який вже і відправляється на сторону клієнта. Отож користувач не має доступу до PHP коду,і це гарантує захищеність виконання скриптів від дій користувача або зловмисників. Це є

перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але JavaScript коди можуть працювати паралельно і незалежно від виконання PHP скриптів на стороні серверу.

JavaScript (JS) — динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. [18] Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

MySQL — система керування реляційними базами даних з ліцензією на вільне використання. Зараз MySQL — дуже поширена СКБД яка отримала підтримку багатьох платформ і мов програмування. Найчастіше вона використовується, для оперативного отримання даних з подальшою генерацією динамічних веб-сторінок, але може використовуватися і в інших цілях.

Apache HTTP-сервер — веб-сервер з ліцензією на вільне використання, який може працювати на усіх найросповсюдженіших операційних системах, UNIX-подібних, Microsoft Windows, та інших. [7]

Web-сервер Apache підтримує багато функцій, велика частина з них реалізовані як скомпільовані модулі, які підключаються за потреби та дають додаткові можливості. Модулі можуть бути різнопланові, забезпечувати підтримку різних мов програмування або схем аутентифікації. Apache може працювати з такими мовами програмування як: Perl, Python, Tcl, PHP.

Сервер Apache підтримує функції віртуального хостингу і тому одна інсталяція Apache може забезпечувати роботу багатьох веб-сайтів на одному хості, при цьому сайти можуть знаходитися на різних доменних іменах.

Apache розроблюється та підтримується спільнотою розробників відкритого програмного забезпечення під керівництвом Apache Software Foundation.

ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Структурно-функціональне моделювання процесу відвантаження рідини

Функціональна модель представляє з необхідним ступенем деталізації систему функцій, які в свою чергу відображають свої взаємовідносини через об'єкти системи. Вона являє собою ієрархію взаємопов'язаних діаграм, кожна з яких представляє підсистему або її окрему компоненту. [17] Вершина цієї структури містить загальний опис системи, який деталізується на наступних рівнях декомпозиції.

Для створення функціональної моделі використано програмний продукт компанії Computer Associates - Erwin Process Modeler. Розглянемо головне призначення інформаційної системи – відвантаження рідкого вантажу. Контекстну діаграму даного процесу представлено на рис. 3.1.

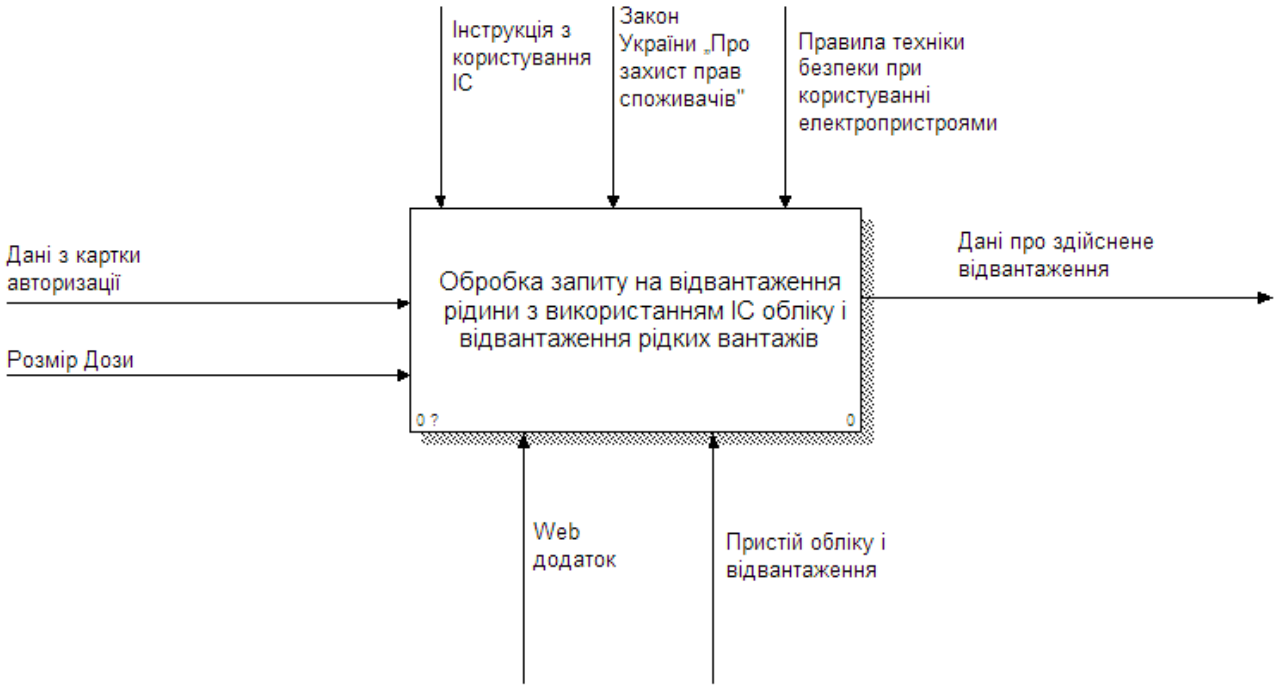

Рисунок 3.1 – контекстна діаграма

Кожна діаграма відображає один чи декілька процесів кожний з яких позначається блоком, також на діаграмах є стрілки, що позначають ресурси, потрібні для виконання функції блоку. [17] Блоки діяльності (activity) є процесом, задачею або функцією які мають визначену назву та виконуються за певний проміжок часу. Діяльність обов'язково моє видавати певний результат, що позначається стрілками направленими з блоку. Діяльність зображується прямокутником, назва якого відображає його функцію у формі дієслова або дієприслівникового звороту. На діаграмі блоки розташовують у порядку домінування з позицій автора. Найменш домінуючий блок розташовується у нижньому правому куті, а найбільш домінуючий розташовують у верхньому лівому. В результаті чого, внутрішня структура діаграми зображує, які функції здійснюють вплив на інші функції. Всі блокам присвоюють певний номер в порядку домінування. Дуги (arrow) зв'язують блоки відображаючи їх взаємодію, та зображують об'єкти або потоки даних. Зв'язки між блоками зображають лініями зі стрілками на кінцях. Імена стрілок можуть приєднуватись до них за допомогою зигзагів (squiggle). В залежності від того з якої сторони в блок входить стрілка дуги можуть позначати такі відношення: вхід (input) – ліва сторона блоку, контроль (control) – верхня сторона, вихід (output) – права сторона та механізм (mechanism) – стрілка яких підходить до блоку знизу.

Вхідними стрілками при обробці запиту на відвантаження є:

- дані з картки авторизації;
- введений розмір дози відвантаження.

На виході ми отримаємо звіт про здійснене відвантаження.

Обробка запиту на відвантаження керується Законом України «Про захист прав споживачів», інструкцією з використання інформаційної системи та правилами безпеки при використанні електропристроїв.

Механізмами є пристрій обліку та інформаційна система.

Будь-яка діяльність діаграми (батьківська) може бути деталізована на діаграмі декомпозиції, яка є дочірньою щодо батьківської діаграми. На діаграмі декомпозиції зображаються блоки діяльності, що представляють функції, які є

складовими батьківської діяльності у процесі детального аналізу. Всі дуги батьківської діяльності автоматично переносяться на дочірню діаграму під час її створення. Для зв'язку батьківської та дочірньої діаграм користуються С-номерами, що дають можливість виключити неоднозначність зв'язку між діаграмами.

На першому рівні деталізації моделі головна батьківська діаграма декомпозується на наступні блоки представлено на рис. 3.2.

- авторизація в системі;
- перевірка параметрів відвантаження;
- відвантаження рідини.

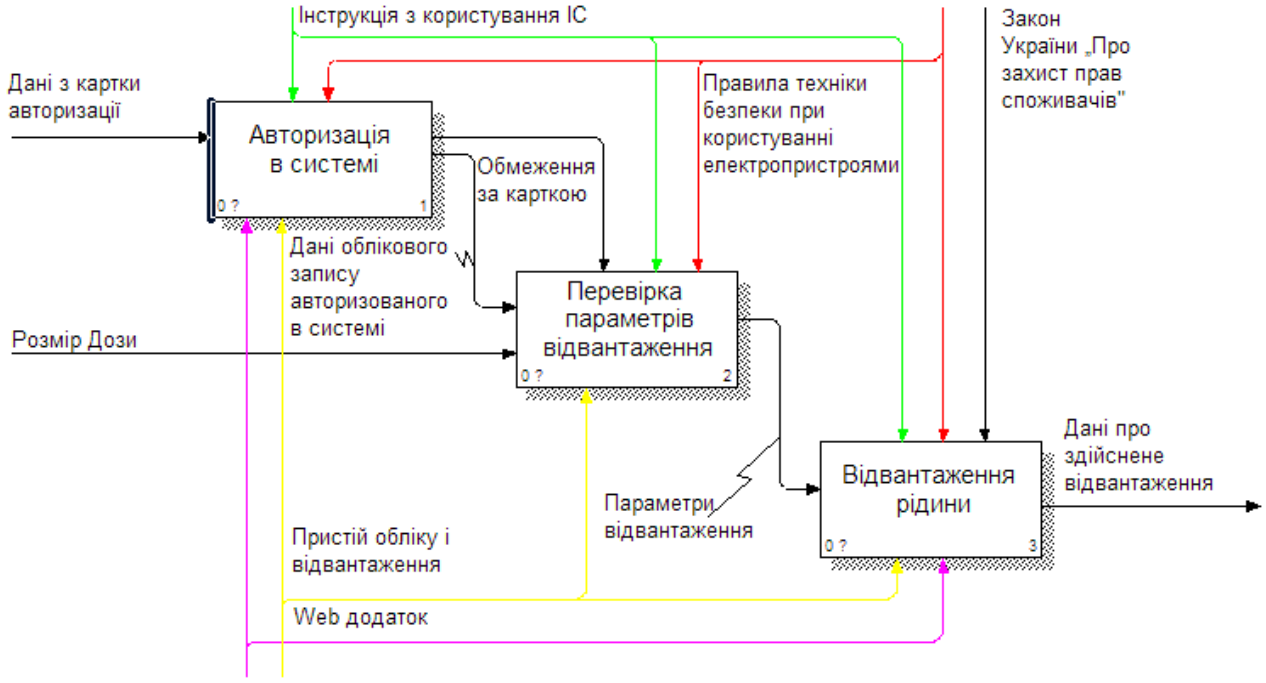

Рисунок 3.2 – Діаграма декомпозиції IDEF0

Вхідними стрілками до діяльності «Авторизація в системі» є «Дані з картки авторизації»; вихідними – «Дані облікового запису авторизованого в системі» та «Обмеження за карткою»; стрілками контролю – «Інструкція з користування ІС» та «Закон України «Про захист прав споживачів»»; стрілками механізмів – «Web додаток» та «Пристрій обліку і відвантаження».

Вхідними стрілками до діяльності «Перевірка параметрів відвантаження» є «Дані облікового запису авторизованого в системі» та «Розмір дози»; стрілками контролю – «Інструкція з користування ІС» та «Закон України «Про захист прав споживачів»» та «Обмеження за карткою»; стрілками механізмів – «Пристрій обліку і відвантаження».

Вхідними стрілками до діяльності «Відвантаження рідини» є «Параметри відвантаження»; вихідними – «Дані про здійснене відвантаження»; стрілками контролю – «Закон України «Про захист прав споживачів», «Інструкція з користування ІС», «Правила техніки безпеки при користуванні електропристроями»; стрілками механізмів – «Web додаток» та «Пристрій обліку і відвантаження».

Декомпозуємо кожний процес на діаграмі А0, першим деталізуємо процес «Авторизації в системі», схему зображено на рис. 3.3.

Рисунок 3.3 – Діаграма декомпозиції IDEF0 «Авторизація в системі»

Вхідними стрілками до діяльності «Зчитування даних доступу з картки» є «Дані з картки доступу»; вихідними – «Дані для перевірки»; стрілками контролю

– «Інструкція з користування ІС» та «Правила техніки безпеки при користуванні електропристроями»; стрілками механізмів – «Пристрій обліку і відвантаження».

Вхідними стрілками до діяльності «Перевірка даних доступу» є «Дані для перевірки»; вихідними – «Дані облікового запису авторизованого в системі»; стрілками механізмів – «Пристрій обліку і відвантаження» та «Web додаток».

Декомпозуємо процес «Перевірка параметрів відвантаження», схему зображено на рис. 3.4.

Рисунок 3.4 – Діаграма декомпозиції IDEF0 «Авторизація в системі»

Вхідними стрілками до діяльності «Введення розмірності відвантаження» є «Дані облікового запису авторизованого в системі» та «Розмір дози»; вихідними – «Розмір дози»; стрілками механізмів – «Пристрій обліку і відвантаження»; стрілками контролю – «Правила техніки безпеки при користуванні електропристроями», «Інструкція з користування ІС».

Вхідними стрілками до діяльності «Перевірка розміру відвантаження» є «Розмір дози»; вихідними – «Параметри відвантаження»; стрілками механізмів – «Пристрій обліку і відвантаження»; стрілками контролю – «Обмеження за карткою».

Декомпозуємо процес «Відвантаження рідини», схему зображено на рис. 3.5.

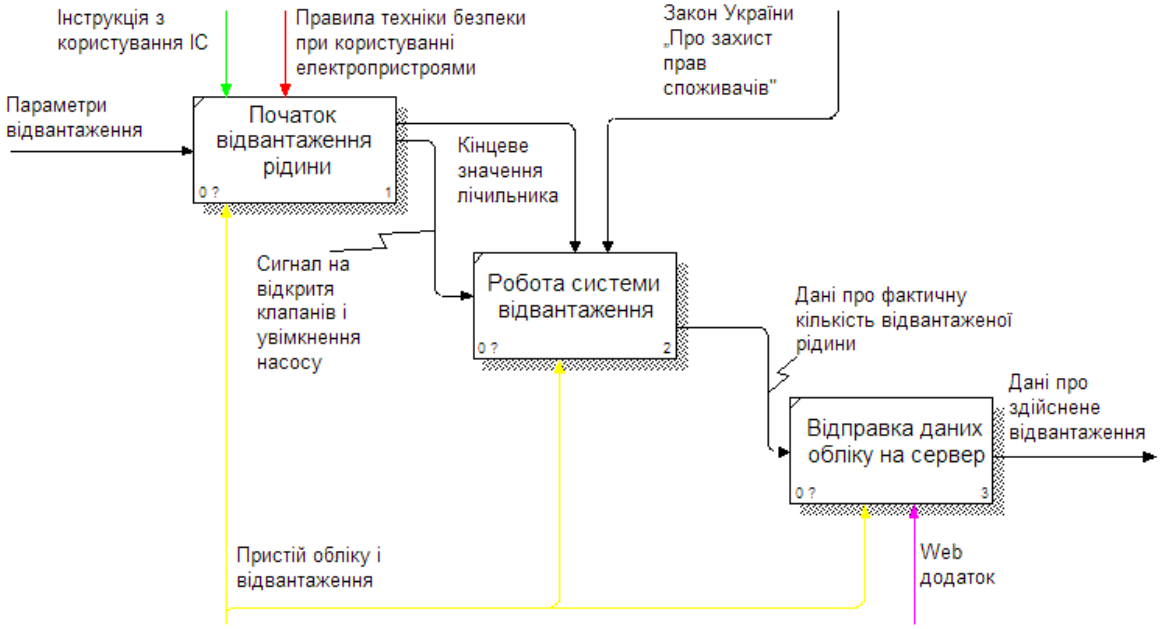

Рисунок 3.5 – Діаграма декомпозиції IDEF0 «Відвантаження рідини»

Вхідними стрілками до діяльності «Початок відвантаження рідини» є «Параметри відвантаження»; вихідними – «Сигнал на відкриття клапанів і увімкнення насосу»; стрілками механізмів – «Пристрій обліку і відвантаження»; стрілками контролю – «Правила техніки безпеки при користуванні електропристроями», «Інструкція з користування ІС».

Вхідними стрілками до діяльності «Робота системи відвантаження» є «Сигнал на відкриття клапанів і увімкнення насосу»; вихідними – «Дані про фактичну кількість відвантаженої рідини»; стрілками механізмів – «Пристрій обліку і відвантаження»; стрілками контролю – «Кінцеве значення лічильника», «Закон України «Про захист прав споживачів».

Вхідними стрілками до діяльності «Відправка даних обліку на сервер» є «Дані про фактичну кількість відвантаженої рідини»; вихідними – «Дані про здійснене відвантаження»; стрілками механізмів – «Пристрій обліку і відвантаження» та «Web додаток».

3.2. Моделювання бази даних інформаційної системи

Реляційна модель даних – це модель даних, де текстова чи числова інформація, що зображається за допомогою таблиць. Кожна таблиця, яка називається відношенням, складається з рядків, які називаються кортежами, та стовпчиків, які називаються атрибутами. Реляційна модель визначає представлення даних (структура), захищеність від некоректних змін (цілісність) та операції, що можуть бути виконані з даними (операції з даними).

ER-діаграми [14] зручні тим, що процес виділення сутностей, атрибутів і зв'язків є ітераційним. Розробивши перший наближений варіант діаграм, вони уточнюються, опитуючи експертів предметної області.

При розробленні інформаційної моделі в термінах ER-моделей ми повинні отримати таку інформацію про предметну область:

- список сутностей предметної області;
- список атрибутів сутностей;
- опис взаємозв'язків між сутностями.

У результаті аналізу предметної області було сформовано перелік функцій які спроектована система повинна виконувати:

- зберігання інформації про пристрої обліку, їх налаштування;
- зберігання інформації про кількість рідини у ємності в пунктах відвантаження;
- зберігання даних про картки доступу до системи;
- ведення журналу про виконані відвантаження;
- забезпечувати доступ до системи користувачам та адміністраторам через web додаток;

Кандидатами на сутність є:

- Пристрій обліку (meter_devices) – явний кандидат на сутність;
- Пункт відвантаження (filling_station) – явний кандидат на сутність;

- Ємність з рідиною (tank) – явний кандидат на сутність;
- Картка (cards) – явний кандидат на сутність;
- Журнал відвантаження (fefill_log) – явний кандидат на сутність;
- Користувач web додатку (user) – явний кандидат на сутність;
- Рідина (type_fluids) – це швидше атрибут;

Визначимо зв’язки між сутностями. Очевидним є зв'язок між сутностями «Ємність з рідиною» та «Пункт відвантаження» - ємність повинна знаходитися на якому небуть пункті відвантаження, а пункт відвантаження має в собі містити одну або більше ємностей.

Наступний зв’язок між сутностями «Пристрій обліку» та «Ємність з рідиною» - пристрій обліку повинен відвантажувати рідину з певної ємності.

Зв’язок між сутностями «Журнал відвантаження» та «Пристрій обліку» - журнал повинен містити в собі дані про пристрій за допомогою якого було здійснено відвантаження.

Зв’язок між сутностями «Журнал відвантаження» та «Картка» - в журналі повинні міститися дані за допомогою якої отримали доступ у систему.

Зв’язок між сутностями «Картка» та «Користувач» - картка повинна бути закріплена за певним користувачем, користувач може мати декілька карток.

Розглянемо властивості(кандидати в атрибути) сутностей.

- Пристрій обліку (ідентифікатор пристрою, IMEI [15] код, номер телефону модему, дата технічного обслуговування, дата та час останньої синхронізації, налаштування ціни імпульсу від витратоміра, номер пломби, залишок коштів на рахунку модема, ідентифікатор під’єднаної емності з рідиною, опис);
- Пункт відвантаження (ідентифікатор пункту відвантаження, назва, довгота, широта, опис);
- Ємність з рідиною (ідентифікатор ємності, максимальна вмістимість, заповненість на даний момент, номер пломби, ціна рідини за куб, ідентицікатор пункту відвантаження, тип рідини, опис);

- Картка (номер картки, дані для перевірки оригінальності картки, номер користувача якому належить картка, ім'я та прізвище користувача картки, обмеження за карткою);
- Журнал відвантаження (номер пристрою обліку, номер картки доступу, об’єм відвантаження, дата та час);
- Користувач web додатку (електронна пошта, пароль, сіль для шифрування паролю, статус користувача);

Після аналізу предметної області переходимо до побудови концептуальної моделі бази даних. Спочатку потрібно визначити сутності на основі аналізу ПО, потім визначити для кожної з них необхідні атрибути, визначити зв’язки та їх характер між сутностями.

Сутність meter_devices(пристрій обліку) має такі атрибути: md_id(ідентифікатор пристрою), md_imei(imei код модему), md_value_of_pulse(налаштування ціни імпульсу від витратоміра), md_date_of_service(дата технічного обслуговування) md_installation_date(дата встановлення), md_seal_number(номер пломби), md_phone_balance(залишок коштів на рахунку модема), md_phone_number(номер телефону модему), md_lust_sync(дата та час останньої синхронізації), md_tn_id(ідентифікатор під’єднаної ємності з рідиною), md_description(опис). На рисунку 3.6 зображена таблиця сутності meter_devices(Пристрій обліку) з усіма зазначеними атрибутами.

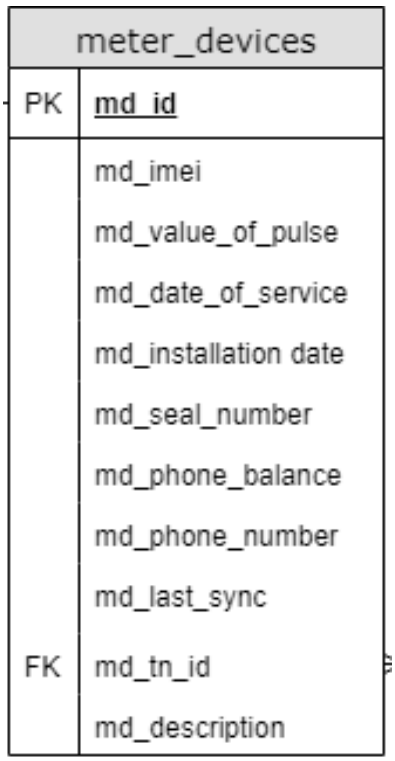

Рисунок 3.6 – Таблиця сутності Meter_devices(Пристрій обліку)

Сутність filling_station(Пункт відвантаження) має такі атрибути: fs_id(ідентифікатор пункту відвантаження), fs_name(назва пункту відвантаження), fs_lat(довгота), fs_lon(широта), fs_description(опис). На рисунку 3.7 зображена таблиця сутності filling_station(Пункт відвантаження) з усіма зазначеними атрибутами.

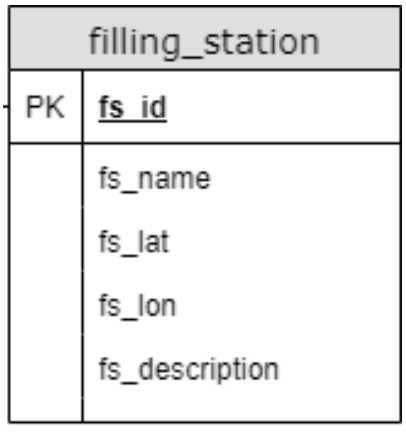

Рисунок 3.7 – Таблиця сутності filling_station(Пункт відвантаження)

Сутність tank(ємність з рідиною) має такі атрибути: tn_id(ідентифікатор ємності), tn_all_capacity(максимальна вмістимість), tn_current_fullness (заповненість на даний момент), tn_seal_number(номер пломби), tn_prise_per_cube(ціна рідини за куб), tn_fs_id(ідентицікатор пункту відвантаження), tn_description(опис), tn_fluid_type(тип рідини). На рисунку 3.8 зображена таблиця сутності tank(ємність з рідиною) з усіма зазначеними атрибутами.

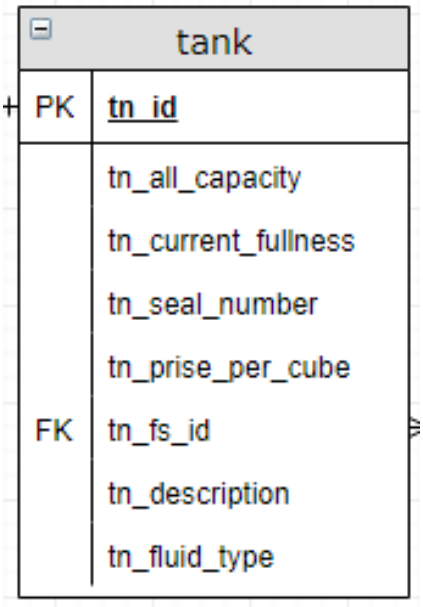

Рисунок 3.8 – Таблиця сутності tank(ємність з рідиною)

Сутність cards(картка доступу) має такі атрибути: cr_number(номер картки), cr_data(дані для перевірки оригінальності картки), cr_user_name(ім'я користувача картки), cr_user_surname(прізвище користувача картки), cr_us_id(ідентифікатор користувача на якого зареєстровано картку), cr_limits(обмеження за карткою), cr_fluid_type(тип рідини). На рисунку 3.9 зображена таблиця сутності cards(картка доступу) з усіма зазначеними атрибутами.

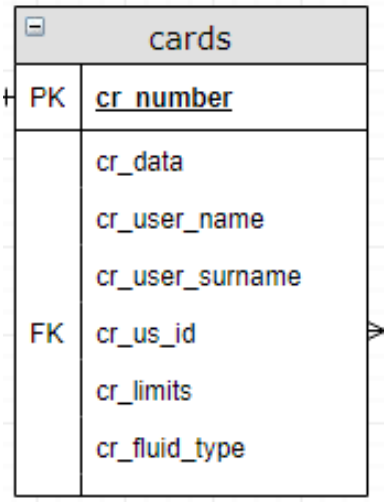

Рисунок 3.9 – Таблиця сутності cards(картка доступу)

Сутність refill_log(журнал відвантажень) має такі атрибути: rl_id(номер запису в журналі), rl_md_id(номер пристрою обліку), rl_cr_number(номер картки доступу), rl_value(об’єм відвантаження), rl_datetime(дата та час відвантаження). На рисунку 3.10 зображена таблиця сутності refill_log(журнал відвантажень) з усіма зазначеними атрибутами.

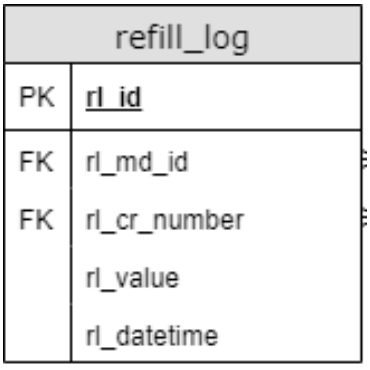

Рисунок 3.10 – Таблиця сутності refill_log(журнал відвантажень)

Сутність users(користувач) має такі атрибути: us_id(ідентифікатор користувача), us_email(електронна пошта), us_password(хеш паролю), us_sold(сіль для хешування), us_role(роль користувача), us_status(статус

користувача). На рисунку 3.11 зображена таблиця сутності users(користувач) з усіма зазначеними атрибутами.

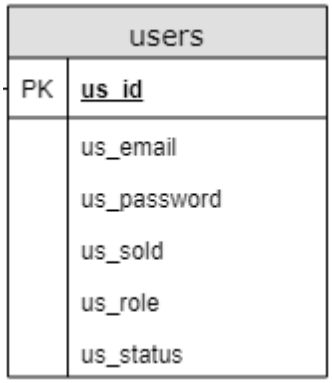

Рисунок 3.11 – Таблиця сутності users(користувач)

Наступним кроком опишемо зв’язки між сутностями.

Сутності filling_station(пункт відвантаження) та tank(ємність з рідиною) мають не обов’язковий зв’язок. Семантика зв’язку - багато до одного. Характеристика зв’язку: Пункт відвантаження має багато ємностей або жодної. Якщо існує ємність з рідиною то вона обов’язково стоїть на пункті відвантаження. На рисунку 3.12 зображено зв'язок цих сутностей.

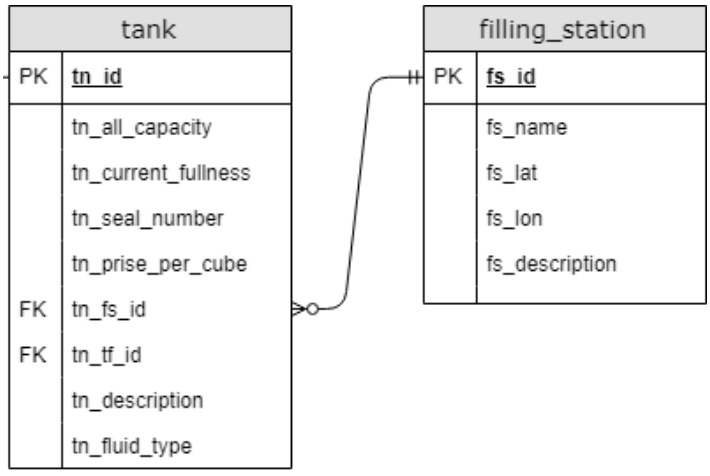

Рисунок 3.12 – Зв’язок сутностей filling_station(пункт відвантаження) та tank(ємність з рідиною)

Сутності meter_devices(пристрій обліку) та tank(ємність з рідиною) мають не обов’язковий зв’язок. Семантика зв’язку - багато до одного. Характеристика зв’язку: До ємності з рідиною підключено один або багато пристроїв обліку. Якщо існує пристрій обліку то він обов’язково підключений до ємності з рідиною. На рисунку 3.13 зображено зв'язок цих сутностей.

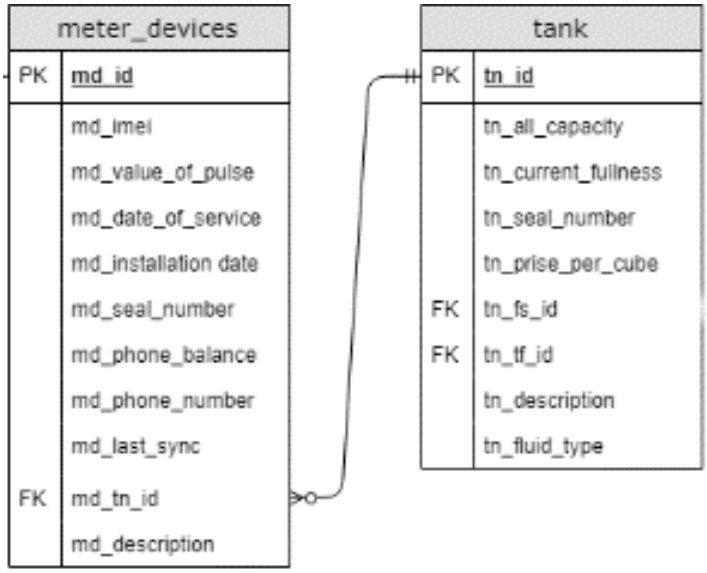

Рисунок 3.13 – Зв’язок сутностей meter_devices(пристрій обліку) та tank(ємність з рідиною)

Сутності meter_devices(пристрій обліку) та refill_log(журнал відвантажень) мають не обов’язковий зв’язок. Семантика зв’язку - багато до одного. Характеристика зв’язку: У журналі відвантажень може бути багато записів з одного пристрою, але якщо є запис він повинен відповідати конкретному пристрою обліку. На рисунку 3.14 зображено зв'язок цих сутностей.

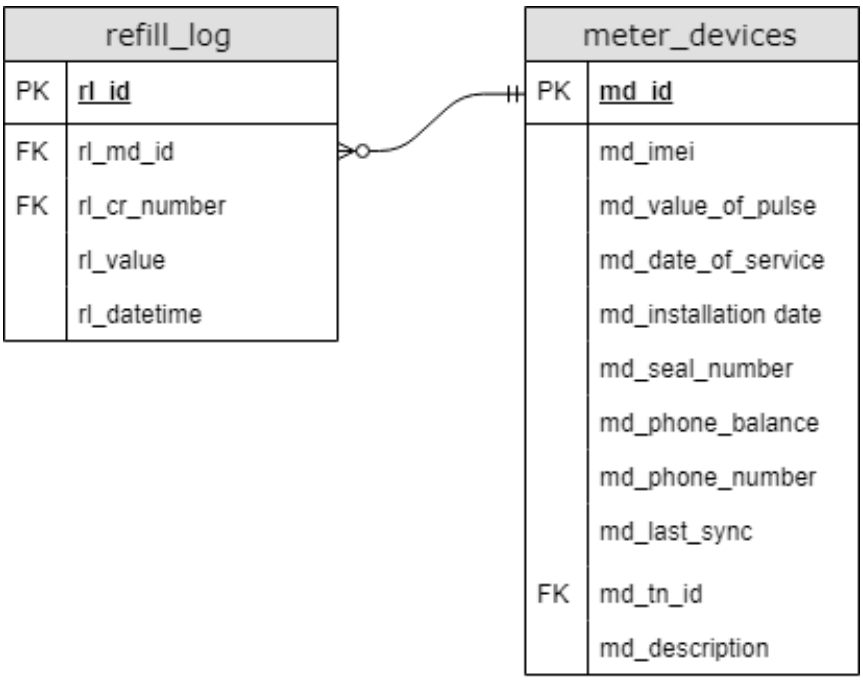

Рисунок 3.14 – Зв’язок сутностей meter_devices(пристрій обліку) та refill_log(журнал відвантажень)

Сутності cards(картка доступу) та refill_log(журнал відвантажень) мають не обов’язковий зв’язок. Семантика зв’язку - багато до одного. Характеристика зв’язку: У журналі відвантажень може бути багато записів про одну і ту саму картку, але якщо є запис він повинен відповідати конкретній картці доступу. На рисунку 3.15 зображено зв'язок цих сутностей.

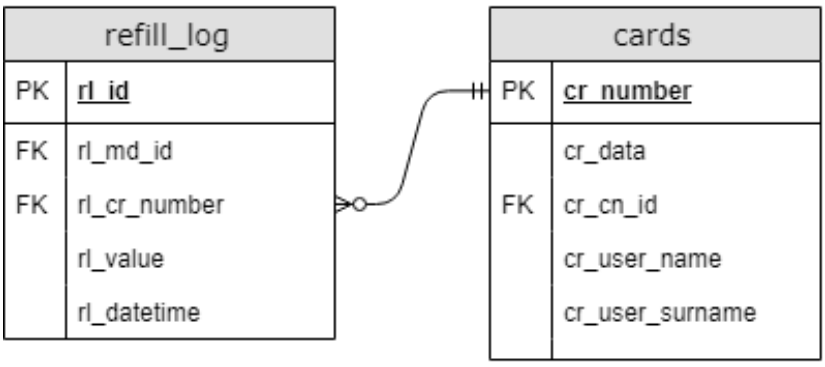

Рисунок 3.15 – Зв’язок сутностей cards(картка доступу) та refill_log(журнал відвантажень)

Сутності cards(картка доступу) та user(користувач) мають не обов’язковий зв’язок. Семантика зв’язку - багато до одного. Характеристика зв’язку: У користувача може бути багато карток, але кожна картка закріплена за конкретним користувачем. На рисунку 3.16 зображено зв'язок цих сутностей.

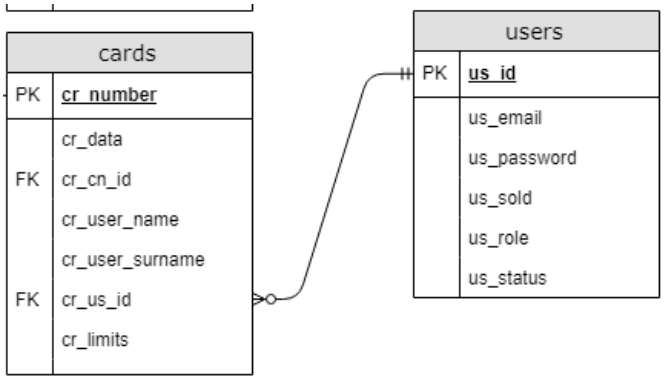

Рисунок 3.16 – Зв’язок сутностей cards(картка доступу) та user(користувач)

Концептуальна модель бази даних інформаційної системи з усіма атрибутами та встановленими зв’язками між об’єктами зображено на рис. 3.17

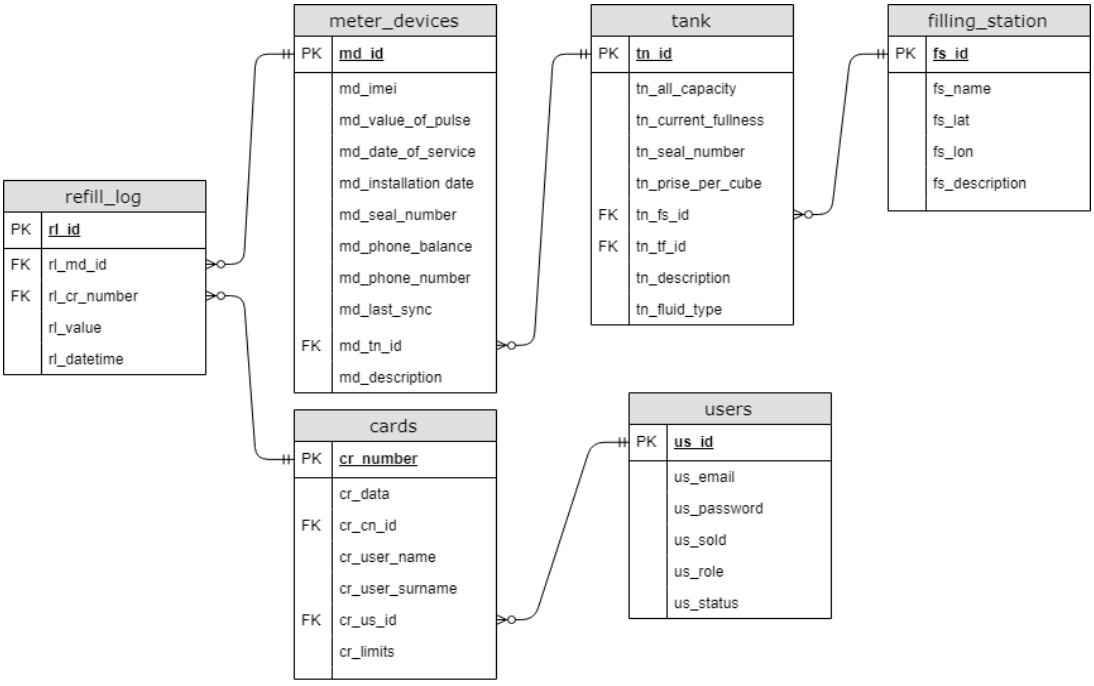

Рисунок 3.17 – Діаграма декомпозиції IDEF0 «Відвантаження рідини»

Розробивши концептуальну модель бази даних, починаємо її аналізувати, аби побудувати логічну модель.

Для зручності використання БД а також з метою зменшення дублювання даних, частину таблиць розбиваємо і створюємо додаткові сутності.

Сутність type_fluids(тип рідини) має такі атрибути: tf_id(ідентифікатор рідини), tf_name(назва рідини), tf_descriptions(опис рідини). На рисунку 3.18 зображена таблиця сутності type_fluids(тип рідини) з усіма зазначеними атрибутами.

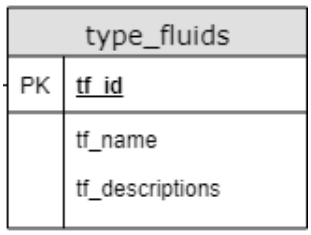

Рисунок 3.18 – Таблиця сутності type_fluids(тип рідини)

Сутність card_limits(обмеження за карткою) має такі атрибути: cl_id(ідентифікатор обмеження), cl_tf_id(номер типу рідини), cl_cr_number(номер картки), cl_balance(залишок), cl_mode(режим обмеження), cl_value(значення обмеження). На рисунку 3.19 зображена таблиця сутності card_limits(обмеження за карткою) з усіма зазначеними атрибутами.

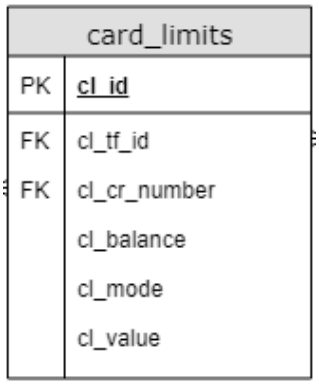

Рисунок 3.19 – Таблиця сутності card_limits(обмеження за карткою)

Сутність consumers(споживач) має такі атрибути: cn_id(номер спаживача), cn_name(назва фірми), cn_phone(контактний телефон), cn_contract_number (номер договору), cn_status(статус), cn_debit(об’єм переплачених послуг), cn_us_id_admin(ідентифікатор користувача адміністратора), cn_description (опис). На рисунку 3.20 зображена таблиця сутності consumers(споживач) з усіма зазначеними атрибутами.

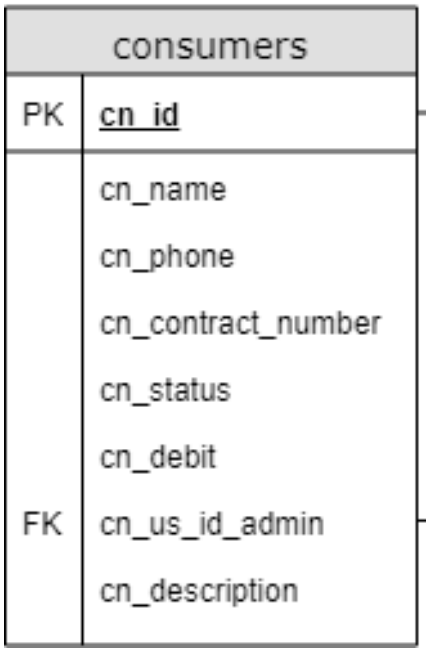

Рисунок 3.20 – Таблиця сутності consumers(споживач)

Сутність change_refill_log(зміни журналу відвантажень) має такі атрибути: crl_id(ідентифікатор запису), crl_rl_id(ідентифікакор запису), crl_rl_md_id(номер пристрою обліку), crl_rl_cr_number(номер картки користувача), crl_rl_value(доза відвантаження), crl_rl_datetime(дата здійснення відвантаження), crl_datetime (дата та час сдійснення зміни в журналі), crl_action_name(тип дії при зміні запису). На рисунку 3.21 зображена таблиця сутності change_refill_log(зміни журналу відвантажень) з усіма зазначеними атрибутами.

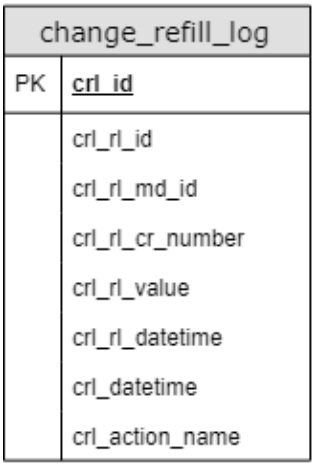

Рисунок 3.21 – Таблиця сутності change_refill_log(зміни журналу відвантажень)

На рисунку 3.22 зображена завершена логічна модель бази даних автошколи з урахуванням змін, що були описані раніше.

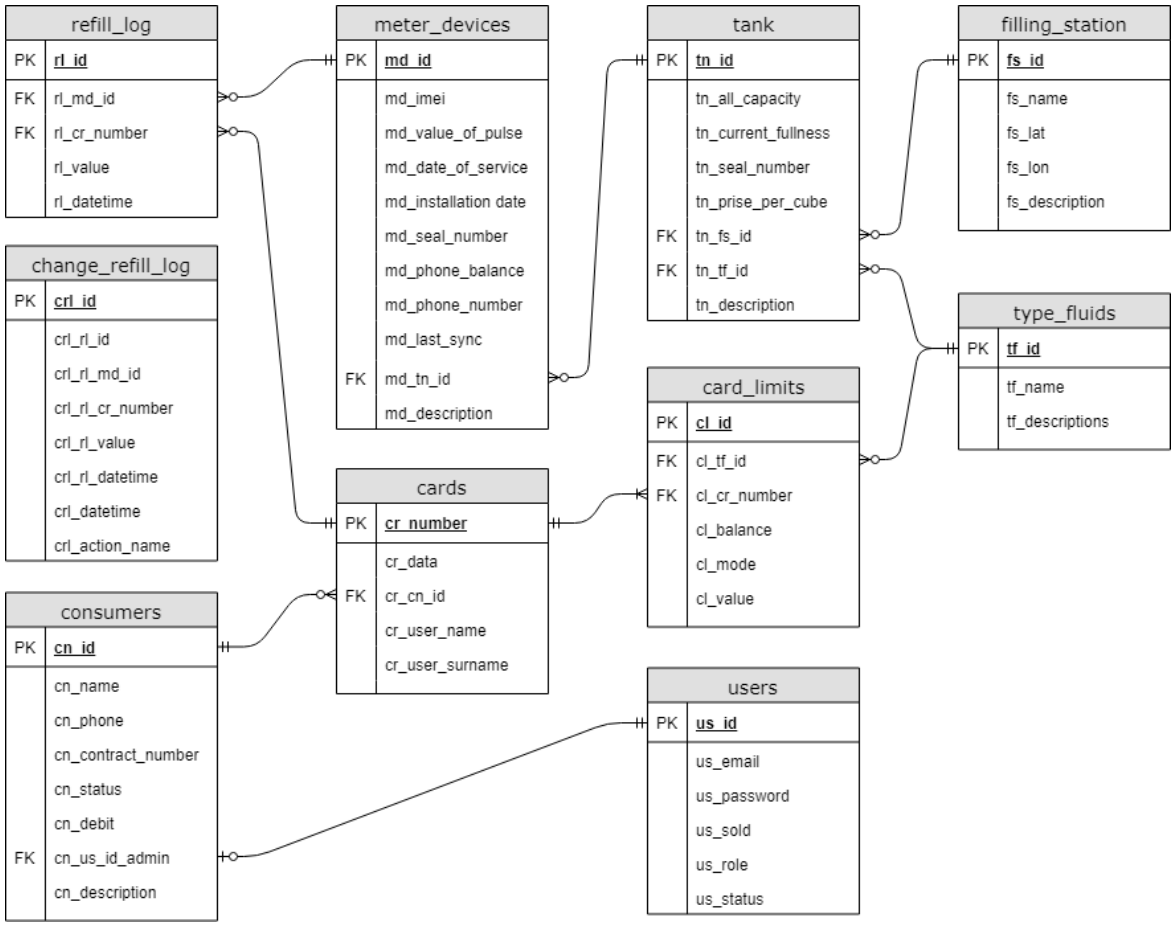

Рисунок 3.22 – Таблиця сутності change_refill_log(зміни журналу відвантажень)

3.3 Моделювання варіантів використання інформаційної системи

Для того, щоб точніше зрозуміти як повинна працювати система, було описано функціональності системи через варіанти використання (Use Case або прецеденти). [19]

Варіанти використання – це опис послідовності дій, які може здійснювати система у відповідь на зовнішні дії користувачів або інших програмних систем. Варіанти використання відображають функціональність системи.

Діаграми варіантів використання описують функціональне призначення системи або те, що система повинна робити. До складу моделі варіантів використання входить опис акторів, варіантів використання (ВВ) програмного продукту (ПП) та діаграми варіантів використання (рис. 3.23).

Актори:

- Користувач пристрою обліку – кінцевий користувач що авторизується в системі і здійснює відвантаження рідини;
- Менеджер компанії споживача – співробітник, який за допомогою web додатку керує картками своєї фірми.
- Адміністратор – співробітник, котрий моніторить роботу системи в цілому, вирішує позаштатні ситуації, коригує дані у разі виникненні помилок, реєструє менеджерів, після укладання договорів, моніторить об’єми рідини в баках у пунктах відвантаження;
- Пристрій обліку – пристрій який здійснює перевірку карток і здійснює облік відвантаженої рідини.

Варіанти використання:

- ВВ Авторизація в Web додатку – ВВ надає права роботи в ІС з певним рівнем доступу;

- ВВ Авторизація в пристрої обліку – ВВ надає право задати дозу для відвантаження, та здійснити відвантаження;
- ВВ Редагування компанії споживача – ВВ дозволяє адміністратору створювати, видаляти, редагувати дані про споживачів;
- ВВ Редагування пункту відвантаження – ВВ дозволяє адміністратору створювати, видаляти, редагувати дані про пункти відвантаження рідини;
- ВВ Редагування ємностей для рідиною – ВВ дозволяє адміністратору створювати, видаляти редагувати записи про ємності для зберігання рідини на пункті відвантаження;
- ВВ Редагування лімітів – ВВ дозволяє менеджеру компанії споживача редагувати ліміти об’єму відвантаження за картками доступу;
- ВВ Редагування карток доступу – ВВ дозволяє менеджеру компанії споживача додавати, видаляти, редагувати картки доступу у систему;
- ВВ Формування звітів – ВВ дозволяє менеджеру компанії споживача формувати звіти за про відвантаження за картками доступу;
- ВВ Додавання звіту про відвантаження – ВВ дозволяє пристрою обліку додавати запис про здійснене відвантаження.

На рисунку 2.23 зображена діаграма варіантів використання.

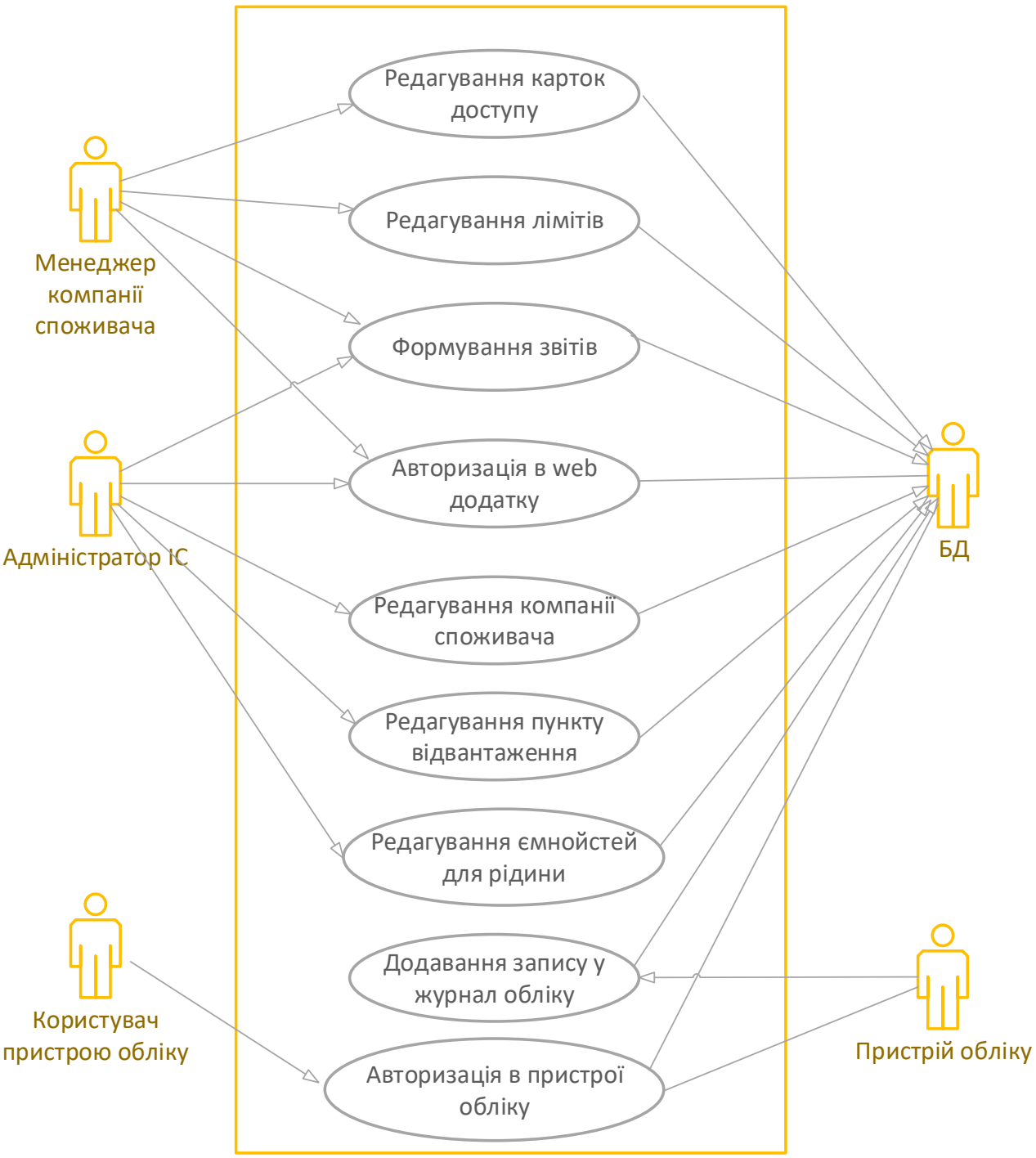

Рисунок 3.23 – Діаграма варіантів використання

4 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Підготовка середовища та розробка компонентів

Web-додаток вирішено розробити як web-сайт. Backend – було вирішено писати самостійно. В якості шаблону розробки було обрано паттерн MVC [13].

Головна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (представлення, виду) [20] рис. 4.1. За рахунок такого поділу підвищується можливість повторного використання коду. Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах. Також при такому підході покращуються зрозумілість структури сайту.

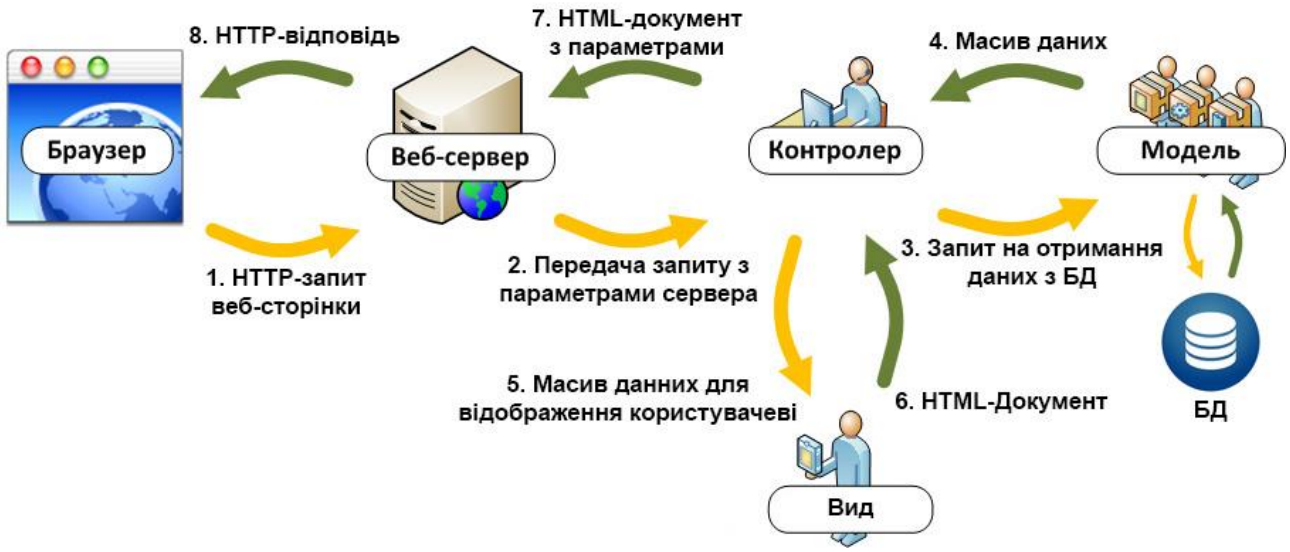

Рисунок 4.1 Структура MVC

При використанні методології MVC можливі такі комбінації:

- до однієї моделі можна приєднати кілька видів, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно

представлені у вигляді електронної таблиці, гістограми і кругової діаграми;

- не змінюючи реалізацію видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних) - для цього досить використовувати інший контролер;

Модель (Model) надає дані і реагує на команди контролера.

Вид (View) відповідає за відображення даних моделі, представлення змінюється при зміни даних моделі.

Контролер (Controller) реагує на дії користувача, передає запити у разі потреби до моделі, завантажує потрібний вид або види.

Серверну частину web додатку було вирішено створити на скриптовій мові програмування - PHP, в якості сервера бази даних обрано MySQL. На рис. 4.2 представлена структура файлів web додатку. Кореневий каталог має в стобі такі файли та папки:

/ application – короневий каталог додатку в цьому зберігаються моделі, види контроллери, та батьківські файли класів, а також файл налаштувань;

/ css – каталог що вістить файли таблиць стилів;

/ fonts – містить файли шрифтів;

/ images – призначена для зберігання зображень;

/ js – директорія JS сценаріїв;

/ vendor – папка для зберігання усіх готових бібілотек, таких як Bootstrap, JQuery, Leaflet та інші;

/ .htaccess – файл правил для сервера Apache;

/ index.php – головна точка входу у створений web додаток.

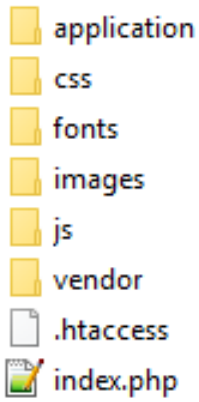

Рисунок 4.2 – Вміст кореневої папки web додатку

Вміст каталогу «application» рис. 4.3:

/ controllers – каталог контроллерів;

/ models – каталог моделей;

/ view – каталог видів;

/ core – директорія містить файли батьківських класів та файл налаштувань;

/ boot.php – файл що підключає основні модулі та виконую маршрутизацію запитів користуваччa;

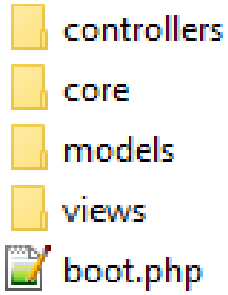

Рисунок 4.3 – Вміст директорії «application»

Щоб створити web-додаток потрібне відповідне оснащення для розробки. Для розробки знадобиться віртуальний хостинг або локальний web сервер.

Оскільки за віртуальний хостинг потрібно сплачувати кошти, на час розробки доцільно використовувати другий варіант. В якості локального web сервера використаємо OpenServer [12], він розповсюджується безкоштовно, та містить в собі усі потрібні для розробки модулі.

Наступним кроком є створення бази даних. База даних була створена у СУБД phpMyAdmin за допомогою MySQL команд, код наведено у ДОДАДКУ В.

PhpMyAdmin дозволяє через браузер здійснювати адміністрування сервера MySQL, запускати запити SQL, переглядати та редагувати вміст таблиць баз даних. Назва створеної бази даних – dose_fluids. На рис. 4.4. зображена створена база даних.

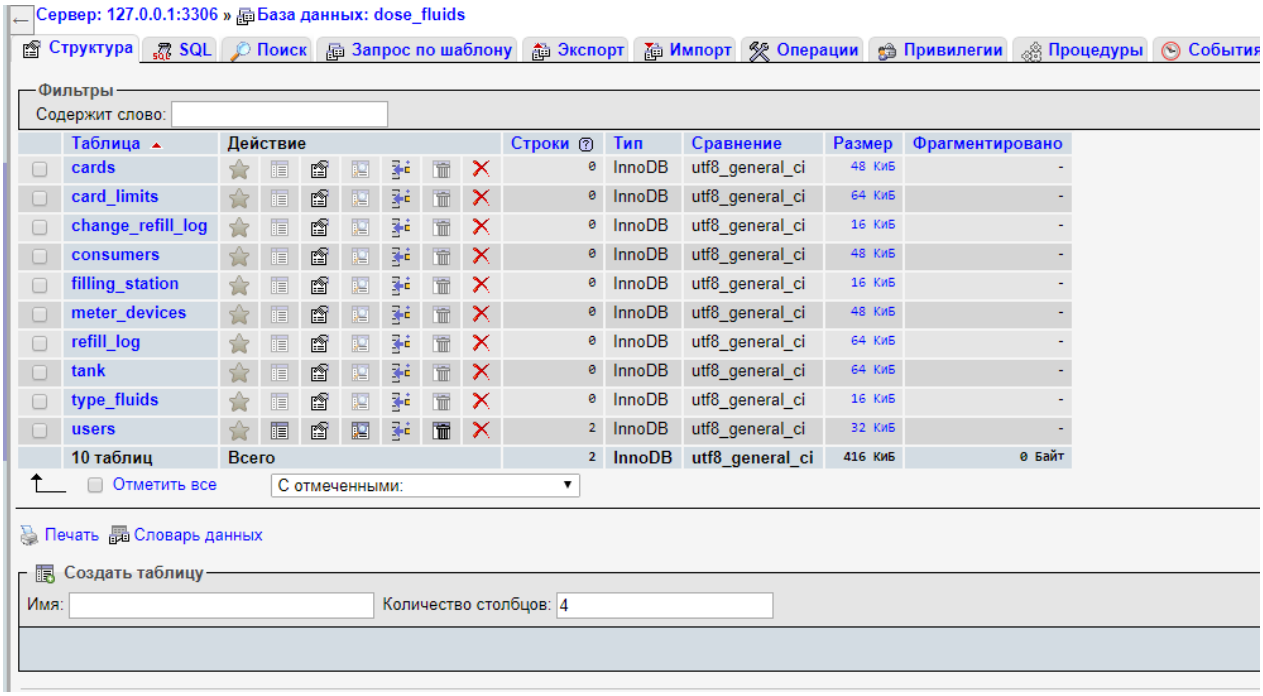

Рисунок 4.4 – База даних в phpMyAdmin

Для підключення створеної БД до web додатку необхідно за допомогою текстового редактору в файл application/core/config.php вписати налаштування підключення до бази даних (рис.4.5).

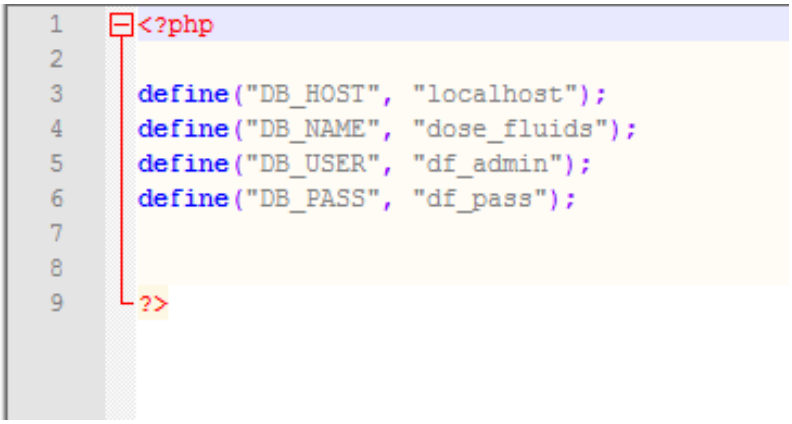

Рисунок 4.5 – Зміст файлу config.php

Клієнтську частину web-сайту було розроблено за допомогою таких інструментів як CSS, HTML, Bootstrap та JavaScript.

Для швидкого створення зручного інтерфейсу web додатку, був використаний Bootstrap [11]. Bootstrap — це фреймворк який включає в себе вже готові модулі та стилі для побудови інтерфейсів користувача, на відміну від серверних модулів він написаний на JavaScript та виконується на стороні користувача.

Bootstrap переважно складається з наборів таблиць стилів LESS та має модульну структуру, таблиці в свою чергу забезпечують роботу різних компонентів та інструментів.

В Bootstrap є такі основні іструменти:

- Сітки (grid) — заздалегідь визначені колонки одразу готові до використання;
- Шаблони (template) — фіксовані чи адаптивні шаблони сторінок;
- Типографіка (typography) — визначенні класи шрифтів, для позначення текстів різного типу і змісту, текст коду, цитати тощо;
- Мультимедіа (media) — засоби для роботи з зображеннями і відео;
- Таблиці (table) — засоби оформлення таблиць, в яких реалізовані такі можливості як сортування;
- Форми (form) — класи для оформлення як форм, так і деяких подій;

- Навігація (nav, navbar) — класи для створення різних варіантів і стилів навігації;
- Сповіщення (alert) — класи для оформлення спливаючих вікон, підказок, діалогових вікон;
- Іконочний шрифт (icon font) — набір іконок у вигляді шрифту.

Для реалізації блоку інтерактивної мапи було обрано JS бібліотеку Leaflet, [16] це безкоштовна бібліотека що дозволяє ставити власні позначки на мапі, додавати до позначок вікна, що розкриваються, накладати маски, створювати зафарбовані фігури на карті, використовувати для карти різні джерела, від карт Google, до власних зображень.

У результаті був реалізований зручний інтерфейс для адміністрування системи. Завдяки тому, що інтерфейс зроблений адаптивним він добре виглядає на великих моніторах рис. 4.6.та 4.7.

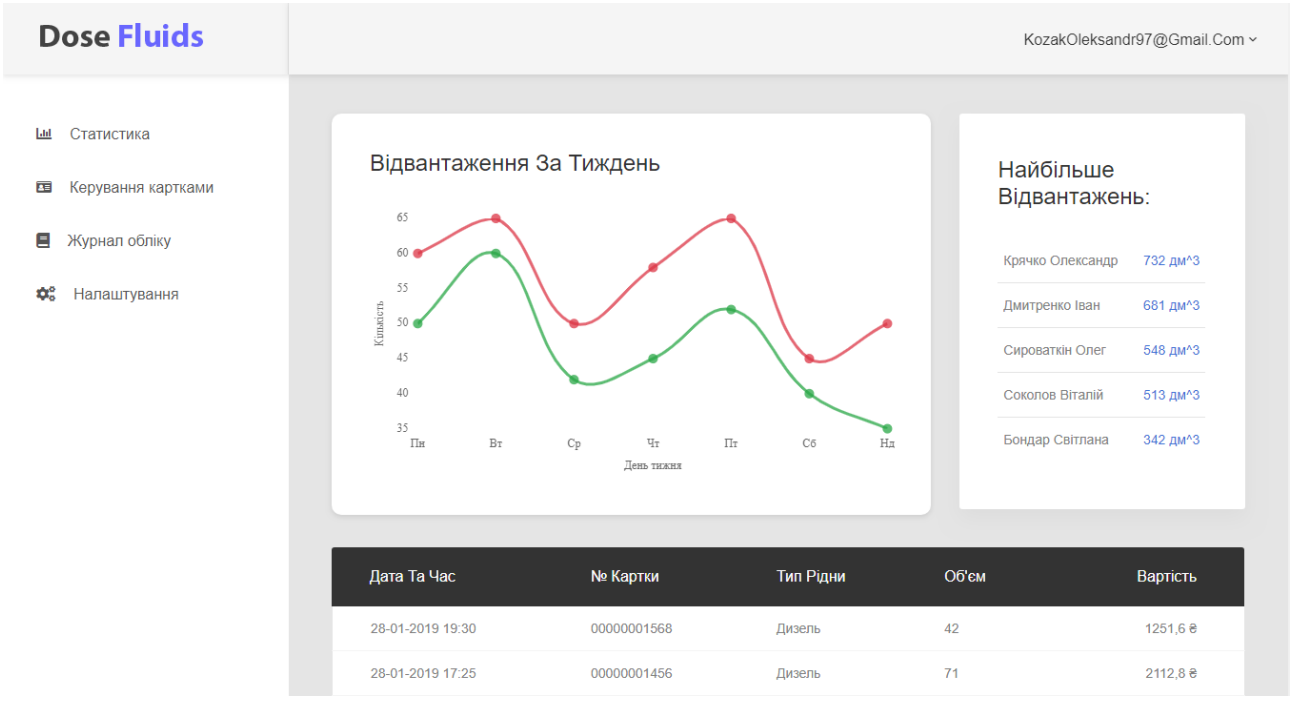

Рисунок 4.6 – Вид на екрані комп’ютера.

А також добре виглядає на екранах мобільних пристроїв рис. 4.7.

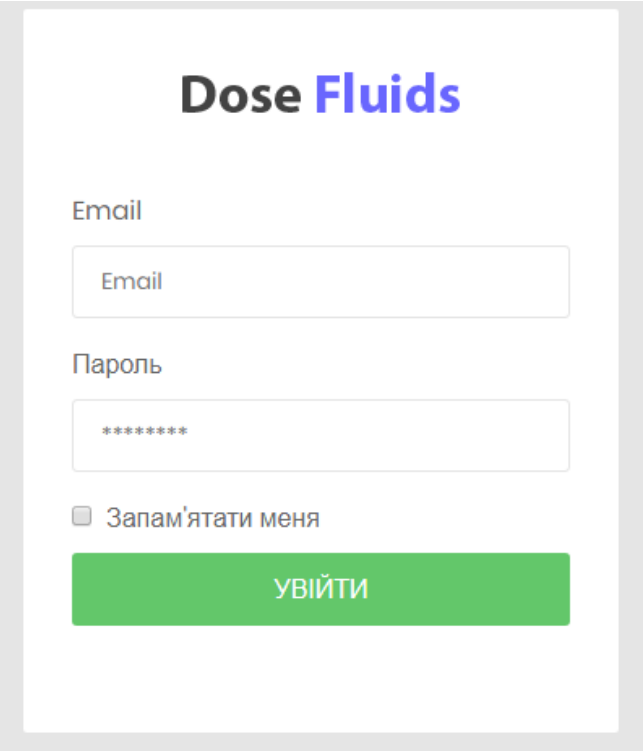

Рисунок 4.7 – Вигляд ІС на екрані смартфону

В версії для смартфонів блоки в правій стороні інтерфейсу розміщуються під блоком меню, один під одним що в значній мірі додає зручності користування з мобільних пристроїв.

4.2. Результат реалізації інформаційної системи

При вводі адреси ІС в браузері користувач потрапляє на сторінку авторизації. На ній можливо ввести логін та пароль для входу до ІС рис. 4.8.

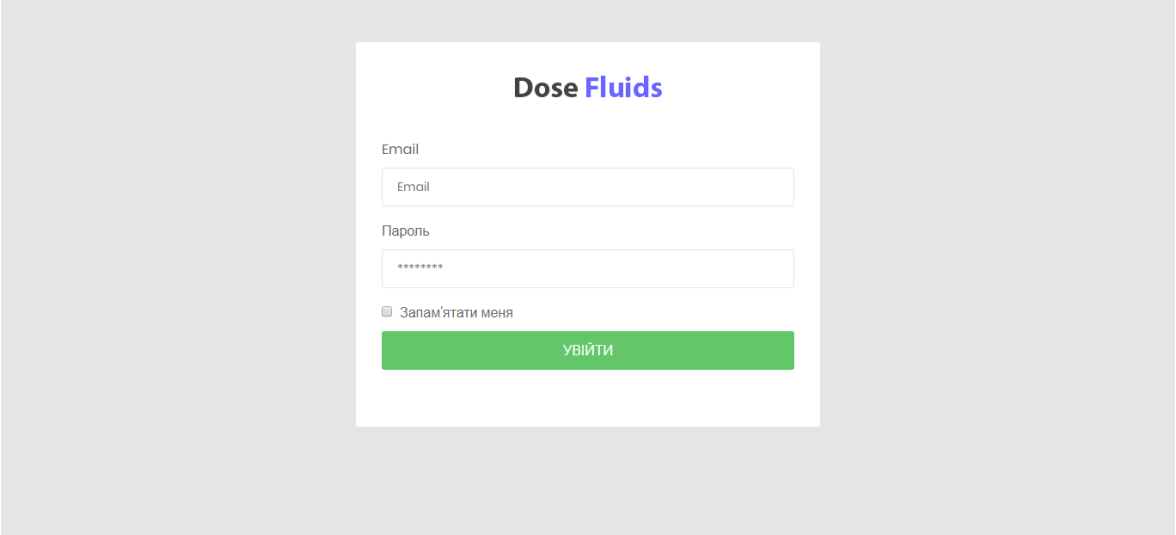

Рисунок 4.8 – Сторінка авторизації

Якщо користувач не зареєстрований, він може зареєструватися натиснувши на текст «Не маю облікового запису» рис. 4.9.

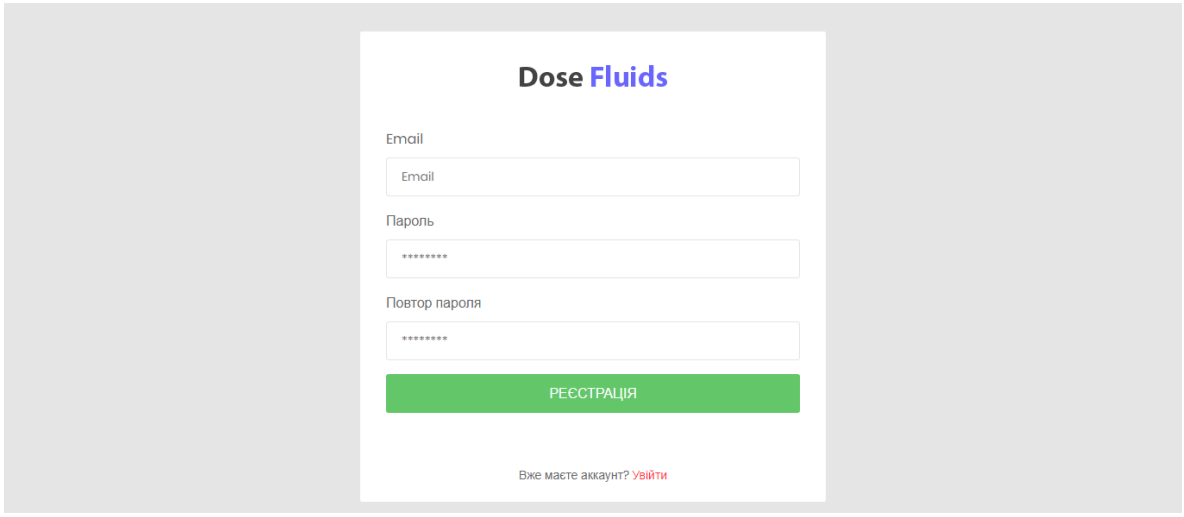

Рисунок 4.9 – Сторінка реєстрації

Після успішної авторизації користувач потрапляє в панель адміністрування, якщо користувач зайшов уперше, йому буде запропоновано заповнити дані про свою компанію рис. 4.10.

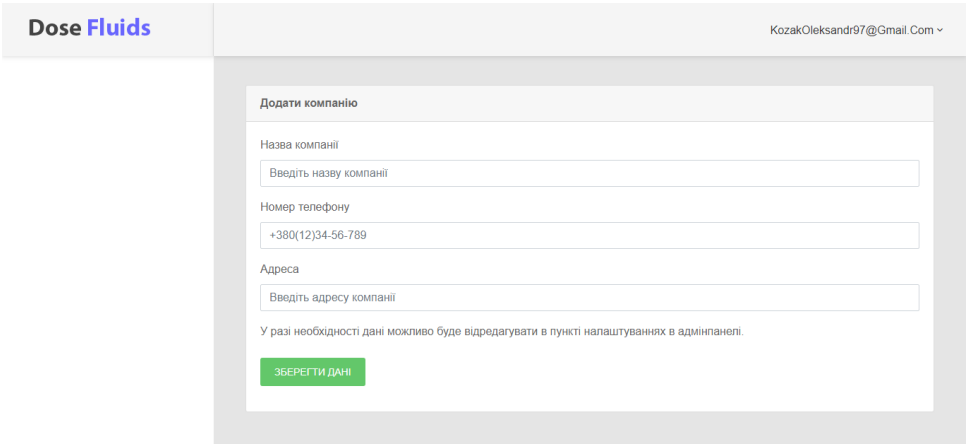

Рисунок 4.10 – Форма заповнення даних про компанію

Після заповнення користувач вже безпосередньо потрапляє в панель керування, образу після входу можна побачити сторінку статисти, вона містить графік відвантажень рідини по дням за останній тиждень, список карт за якими за останній місяць було здійснено найбільше відвантажень, журнал з 10 останніх відвантажень, форму зворотнього зв’язку з адміністратором системи рис. 4.11.

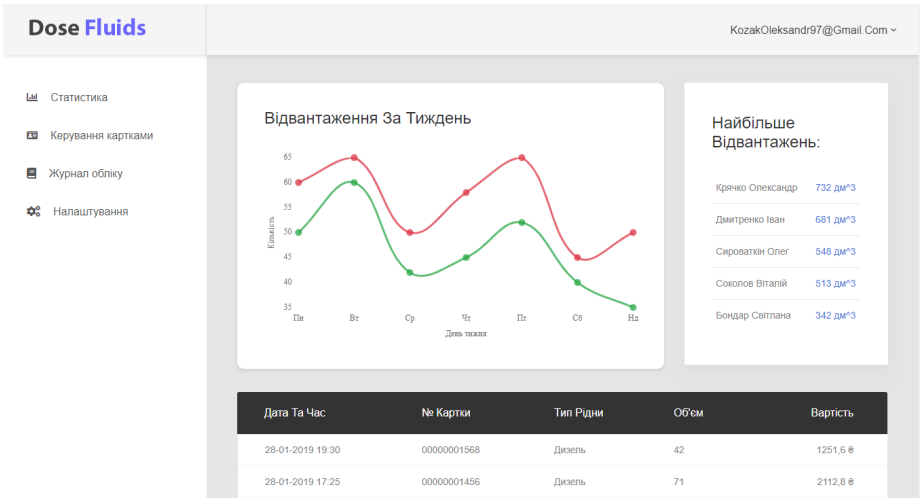

Рисунок 4.11 – Панель керування користувача

В лівій частині екрану є меню при натисканні на пункт «Керування картками», ми потрапляємо на сторінку де можливо адмініструвати картки доступу до системи рис. 4.12. На цій сторінці ми можемо бачити за якою карткою скільки ліміту використано, та об’єм усіх відвантажень за цією карткою.

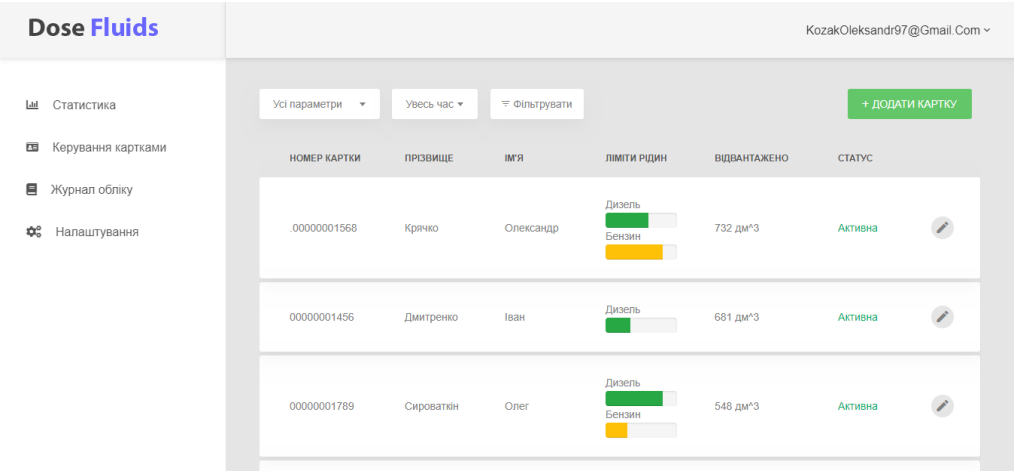

Рисунок 4.12 – Сторінка адміністрування карток

При натисканні кнопки редагувати відкривається вікно редагування картки, тут можливо задати ім’я користувача та ліміти, також є кнопка видалення картки рис. 4.13.

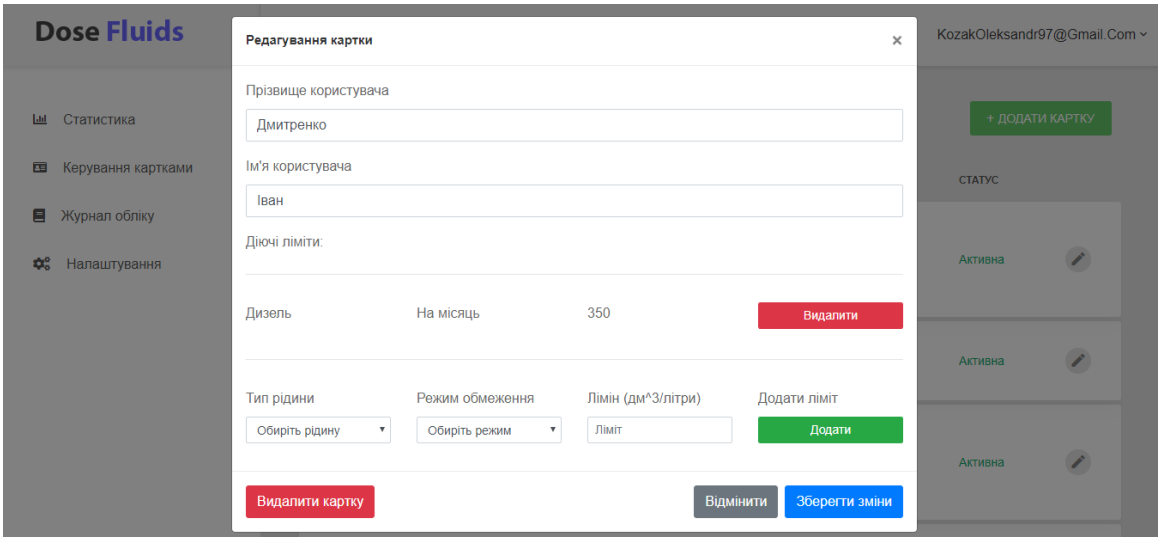

Рисунок 4.13 – Вікно редагування картки

У верхній частині сторінки керування картками є кнопка додати картку, при натисканні цієї кнопки відкривається вікно додавання нової картки рис. 4.14, заповнивши усі поля і натиснувши кнопку «Зберегти» картку буде додано до системи .

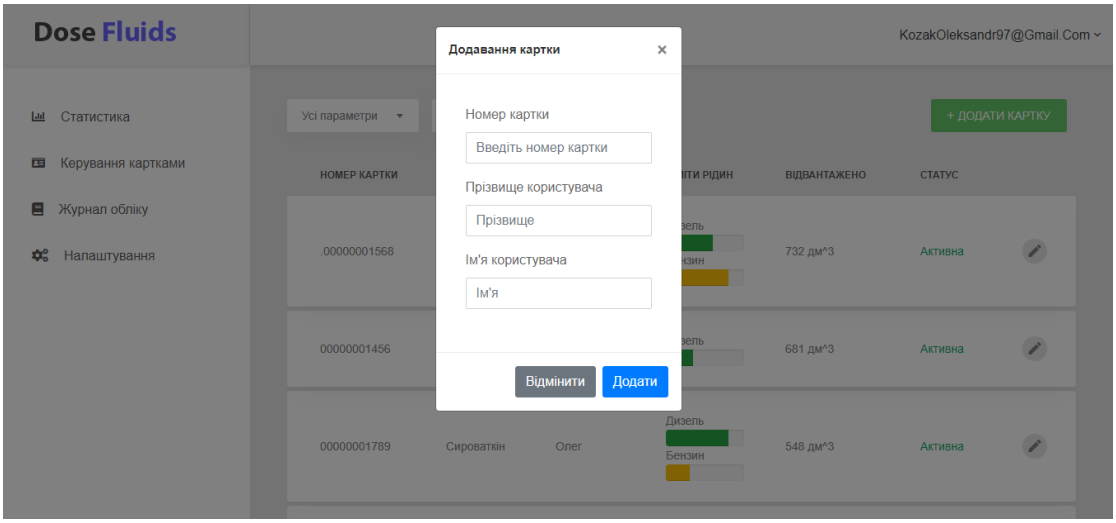

Рисунок 4.14 – Вікно додавання картки

При натисканні пункту меню «Журнал обліку» користувач потрапляє на сторінку де відображаються усі відвантаження за всіма картками, при необхідності в верхній частині можна виконати сортування рис. 4.15.

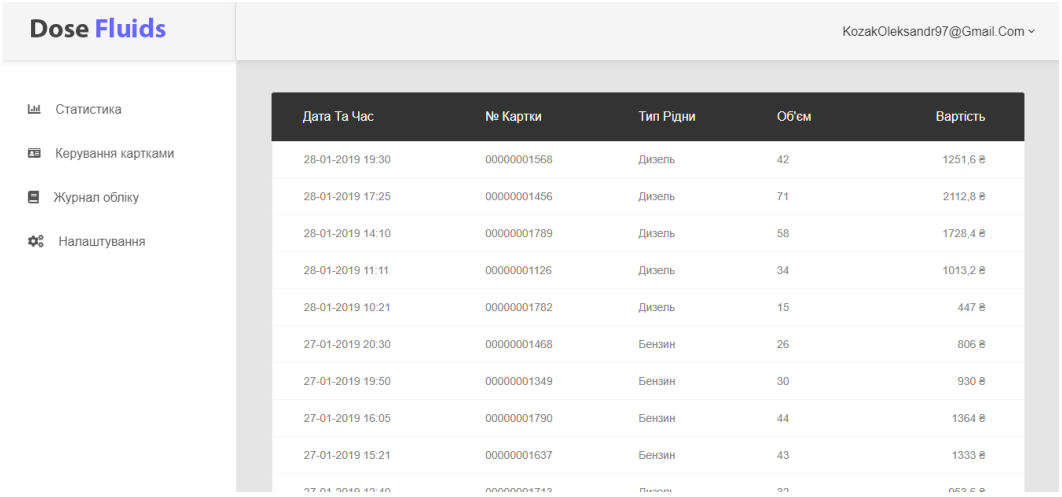

Рисунок 4.15 – Сторінка журналу обліку

На сторінці «Налаштування» можна змінити відомості про компанію а також змінити пароль облікового запису рис. 4.16.

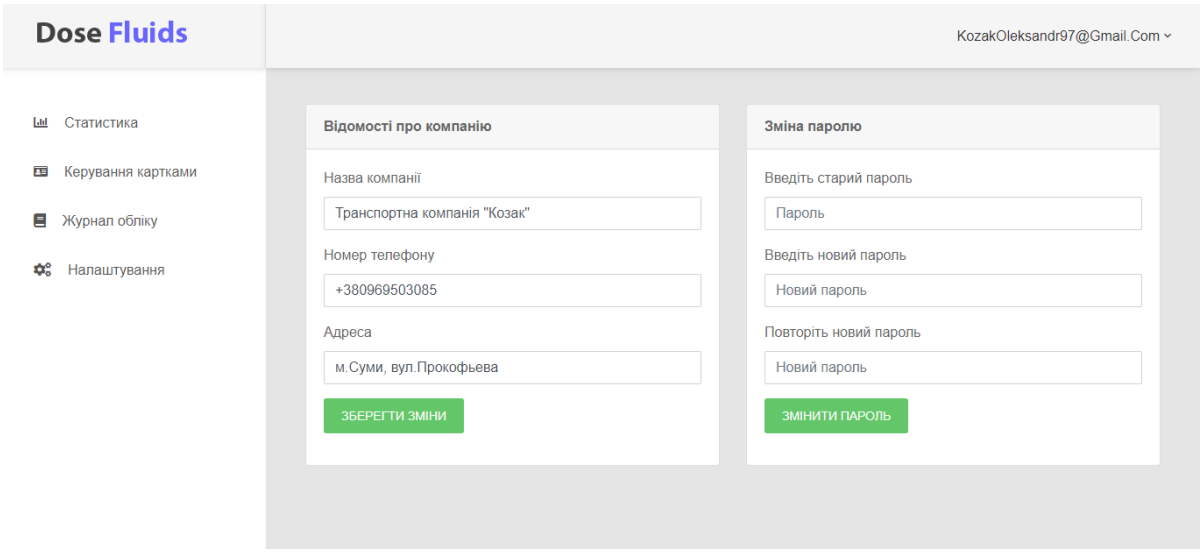

Рисунок 4.16 – Сторінка налаштувань

Авторизація адміністратора системи виконується таким самим шляхом як і авторизація звичайного користувача. При вході в систему адміністратор також бачить сторінку статистики рис. 4.17.

На ній адміністратор може побачити графік відвантажень за всіма пунктами відвантаження за останній тиждень, список пунктів відвантаження в яких ємності з рідиною заповнені менше ніж на 20%, список пунктів відвантаження з найбільшою кількістю відвантажень за останній тиждень, список пристроїв обліку що не виходять на зв’язок, список пристроїв обліку баланс на яких менше 5 гривень.

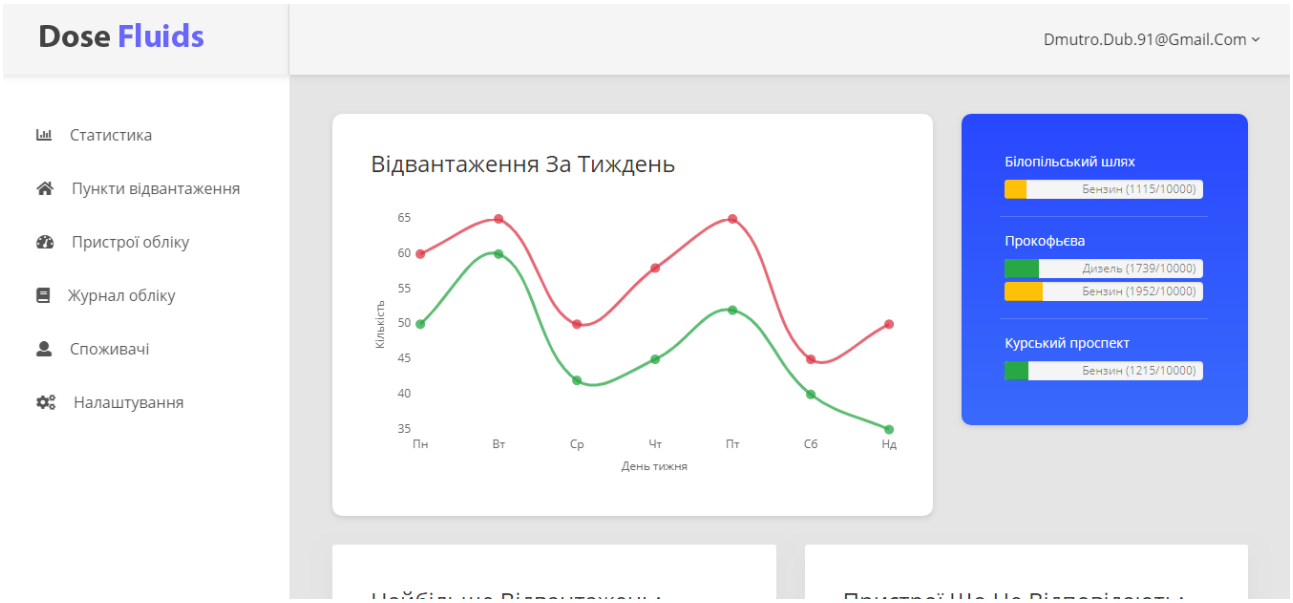

Рисунок 4.17 – Сторінка статистики для адміністратора

В лівій частині у адміністратора є меню з такими пунктами «Пункти відвантаження», «Пристрої обліку», «Журнал обліку», «Споживачі». При переході за пунктом меню «Пункти відвантаження» адміністратор потрапляє на сторінку яка має два режиму, перший це у вигляді таблиці рис. 4.18 та у вигляді мапи з позначками рис. 4.19.

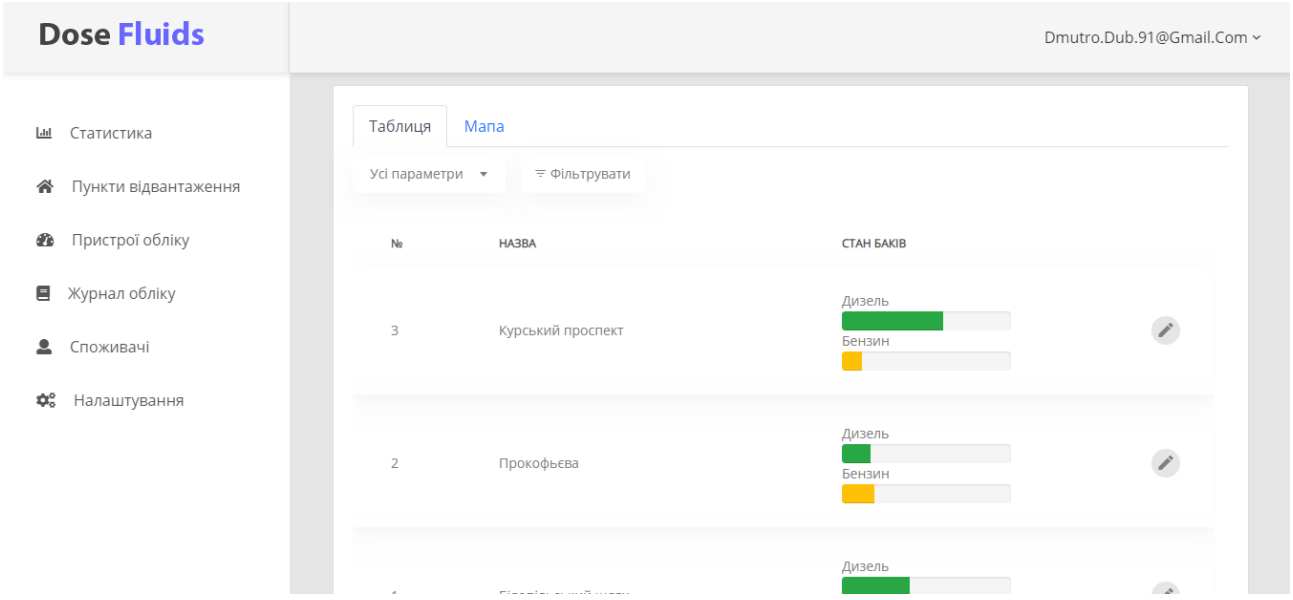

Рисунок 4.18 – Сторінка пунктів відвантаження таблицею.

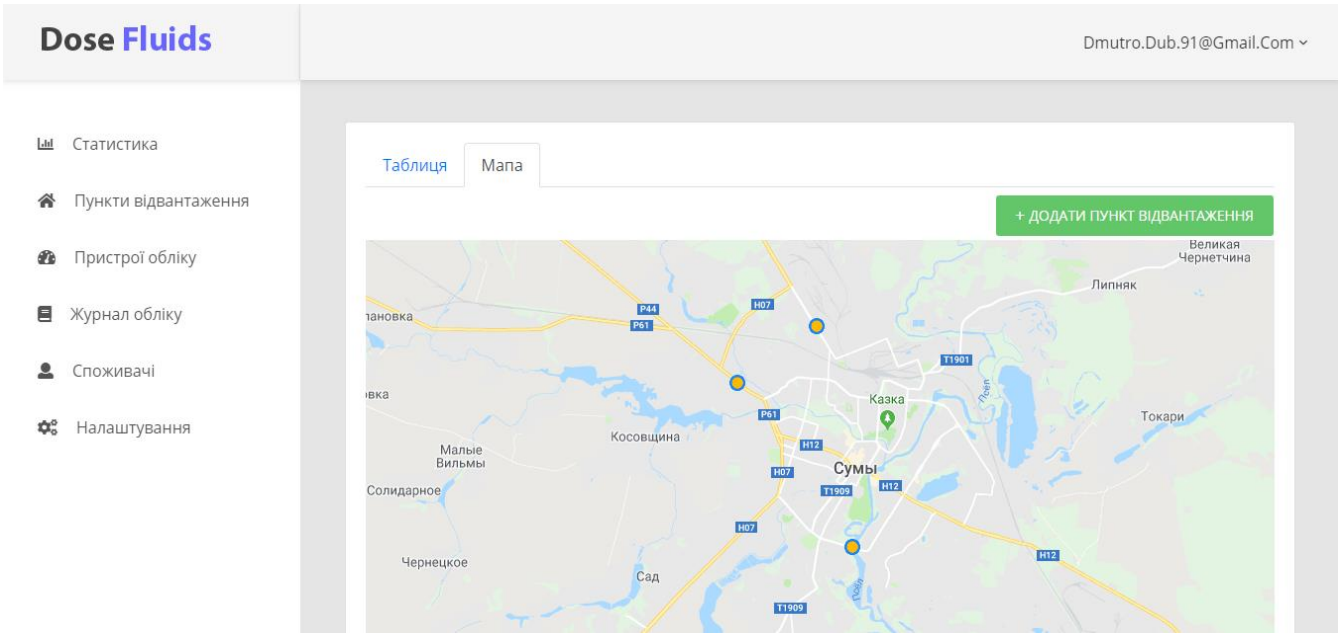

Рисунок 4.19 – Сторінка пунктів відвантаження мапою

Якщо у вигляді таблиці натиснути на кнопку «Редагувати» у рядку відповідного пункту відвантаження то відкриється вікно редагування пункту відвантаження, тут можливо додати ємність, змінити назву пункту обліку, або видалити його рис. 4.20.

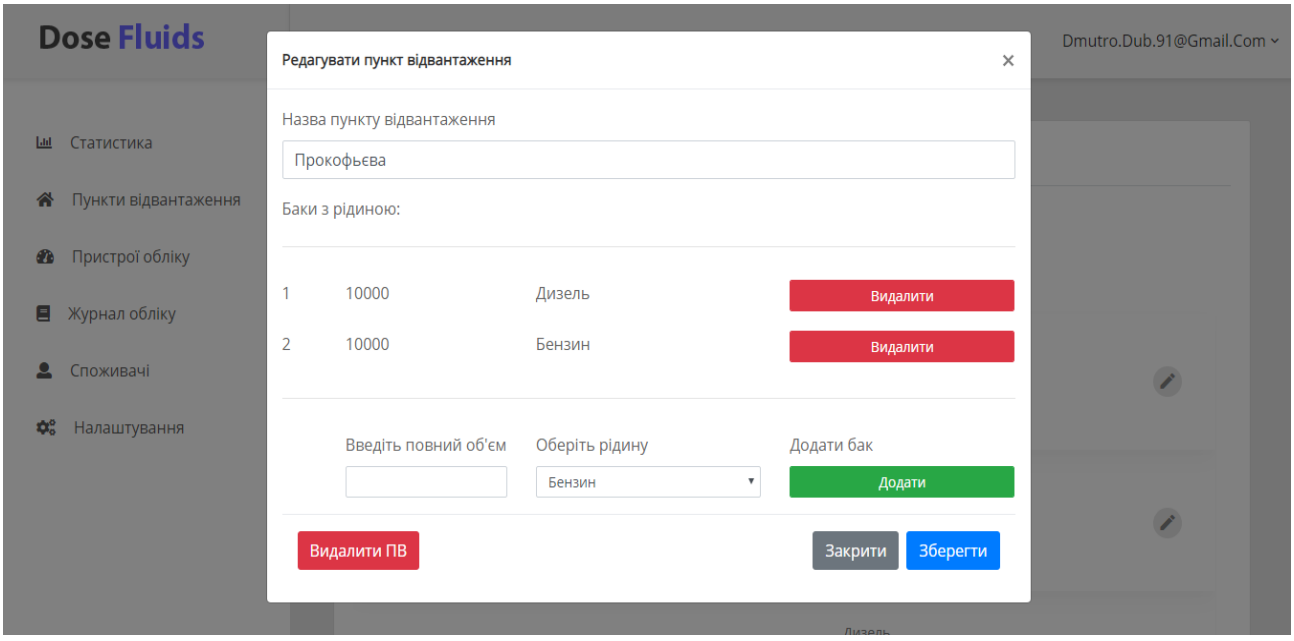

Рисунок 4.20 – Вікно редагування пункту відвантаження

Якщо увійти в режим мапи то ми бачимо мапу з позначками пунктів відвантаження, при натисканні на позначку відкриється віконце з назвою та id пункту обліку, також тут є дві кнопки «Редагувати» та «Змінити місцезнаходження» рис. 4.21, натискання кнопки редагувати призведе до відкриття вікна редагування як на рис. 4.20.

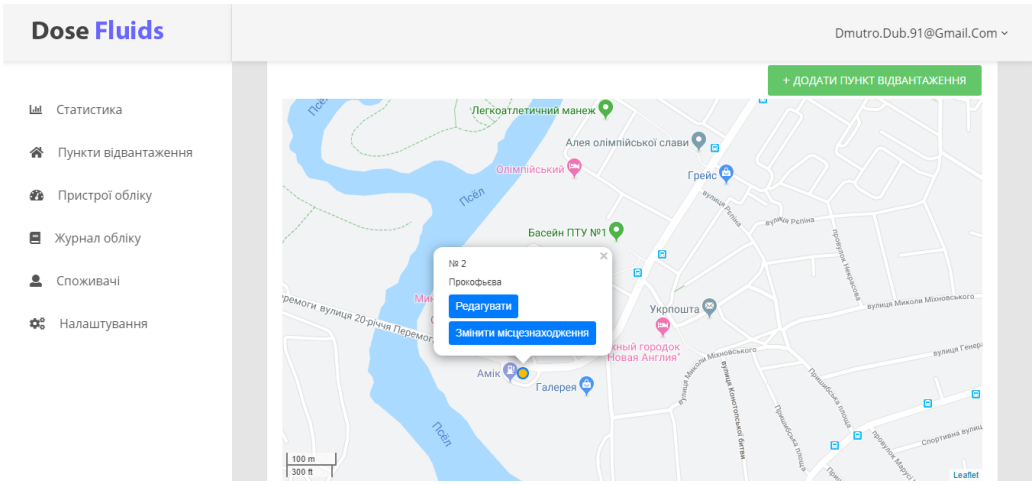

Рисунок 4.21 – Віконце пункту відвантаження на мапі

При натисканні кнопки «Змінити місцезнаходження» стає можливо змінити положення позначки на мапі, після зміни позиціонування треба натиснути кнопку зберегти що розміщено згори над мапою рис. 4.22.

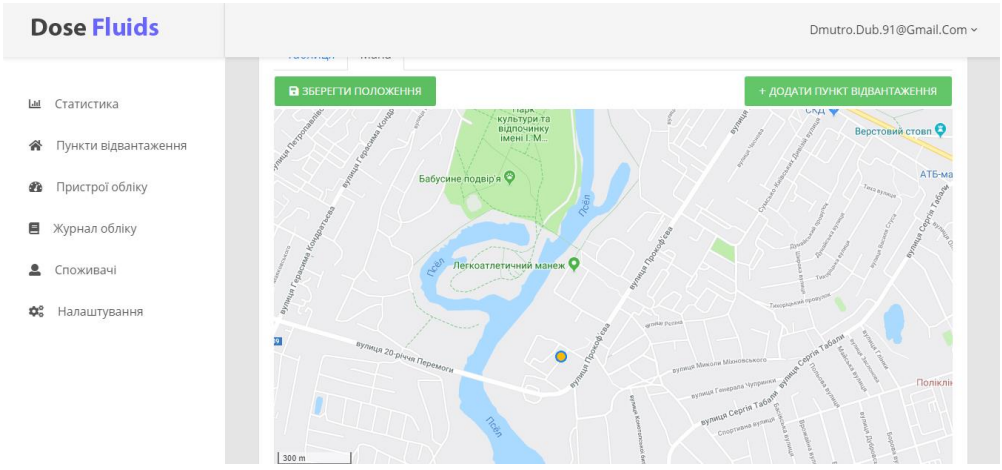

Рисунок 4.22 – Зміна місцезнаходження пункту відвантаження

Також над мапою є кнопка «Додати» при натисканні цієї кнопки стає можливо додати нову мітку рис. 4.23, після позиціонування треба натиснути кнопку зберегти. Після збереження новий пункт відвантаження можливо редагувати так само як і інші, додавати ємності та ін.

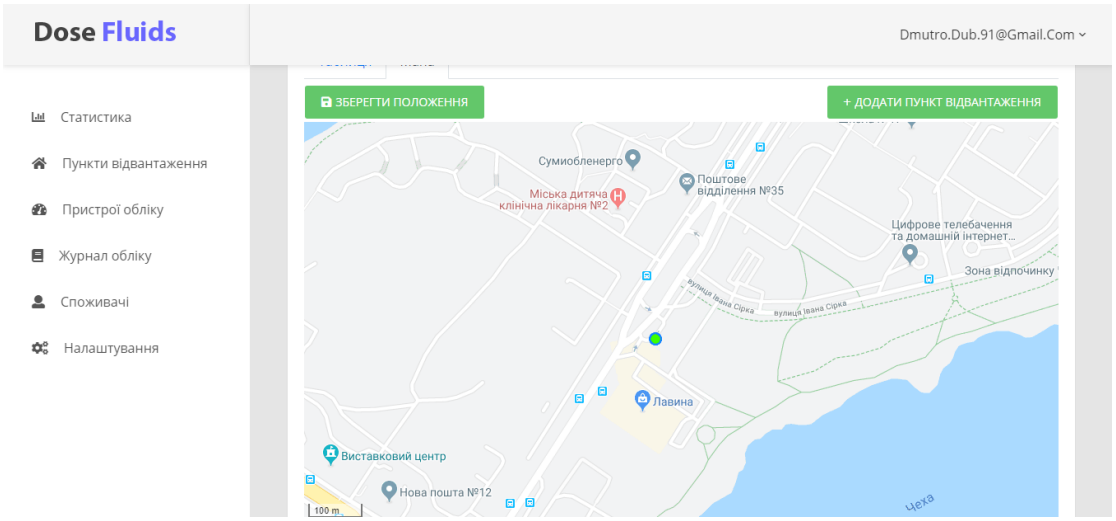

Рисунок 4.23 – Додавання нового пункту відвантаження

При натисканні пункту меню «Пристрої обліку» ми потрапимо на сторінку де міститься інформація про усі пристрої обліку у вигляди таблиці рис 4.24.

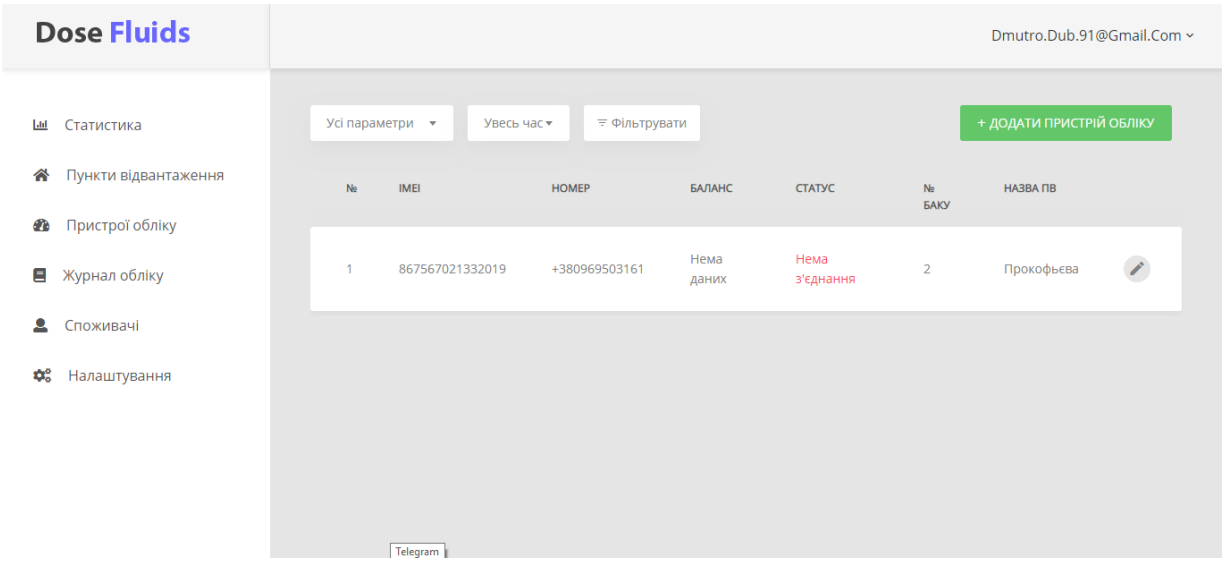

Рисунок 4.24 – Сторінка інформації про пристрої обліку

При натисканні кнопки «Редагувати» у відповідному рядку, відкриється вікно редагування пристрою обліку рис. 4.25.

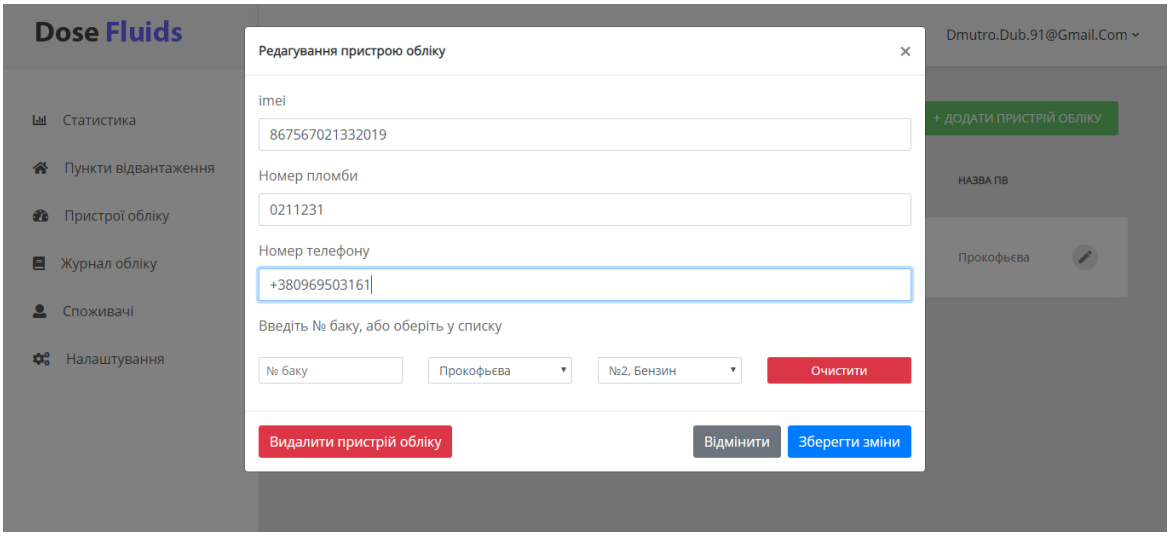

Рисунок 4.25 – Вікно редагування пристрою обліку

При натисканні на пункт меню «Журнал обліку» відкриється сторінка де ми бачимо журнал відвантажень за всіма пунктами обліку, за потреби можливо натиснути кнопку «Видалити» у відповідному рядку, також у верхній частині можливо обрати сортування, рис. 4.26.

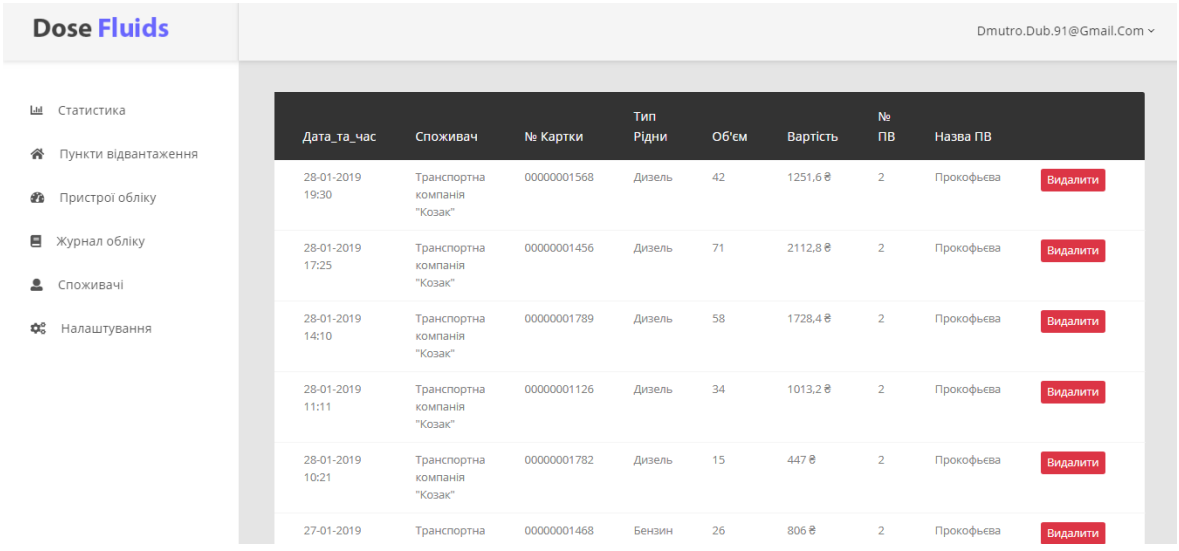

Рисунок 4.26 – Сторінка журналу обліку

При натисканні пункту «Споживачі» буде виведено таблицю з усіма зареєстрованими споживачами у стовпчику статус з випадаючого списку, можливо вибрати статус для компанії, після вибору параметру з’явиться вікно з запитом на підтвердження змін рис. 4.27.

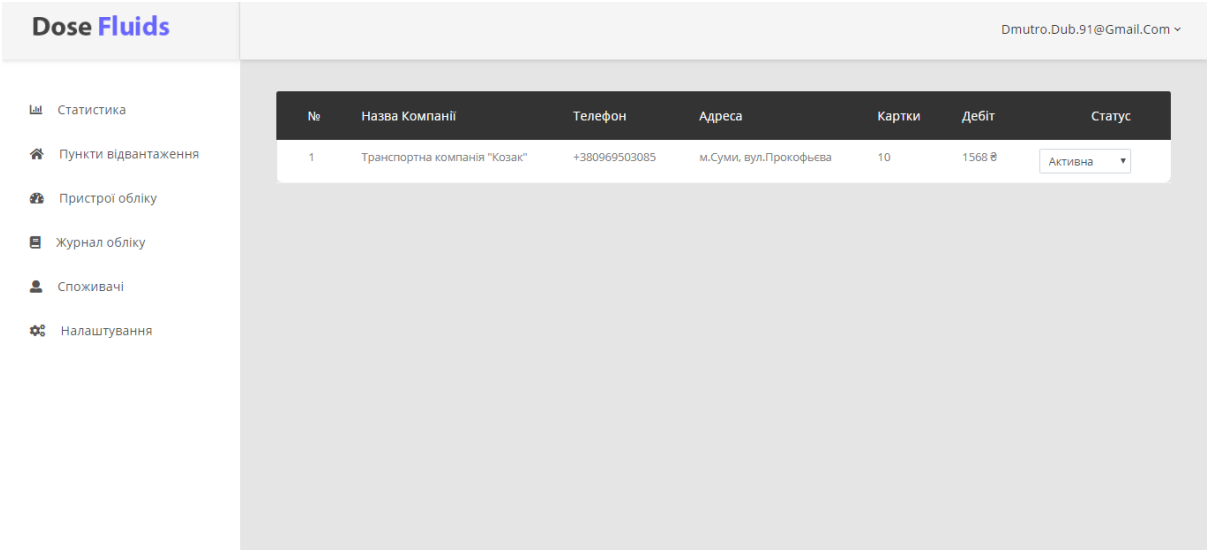

Рисунок 4.27 – Сторінка інформації про споживачів

Для обліку рідини в системі використовуються пристрої обліку рис. 4.28.

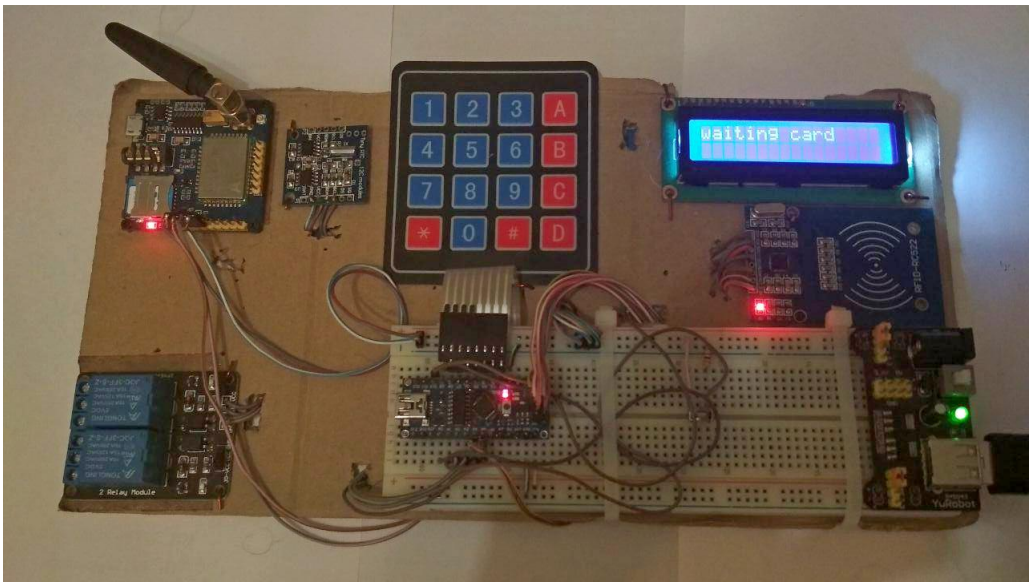

Рисунок 4.28 – Пристрій обліку

Для того щоб почати роботу з ним, треба прикласти картку доступу до зчитувача, після успішної авторизації на екрані з’явиться напис «allowed», а після нього «enter dose» рис. 4.29.

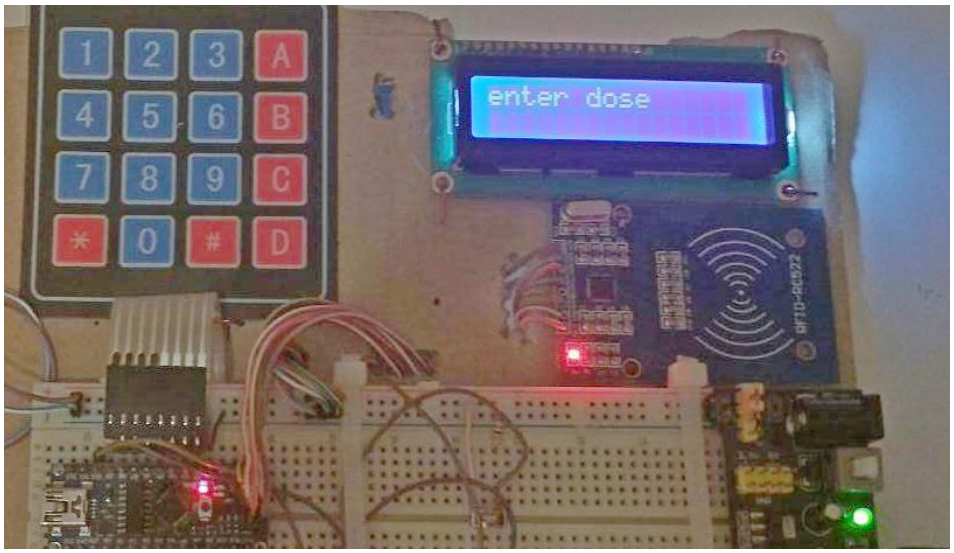

Рисунок 4.29 – Запит на введення дози відвантаження

Для продовження треба ввести розмір дози за допомогою клавіатури, та натиснути кнопку «A», після чого розпочнеться відвантаження, а на екрані з'явиться напис «processed» рис. 4.30.

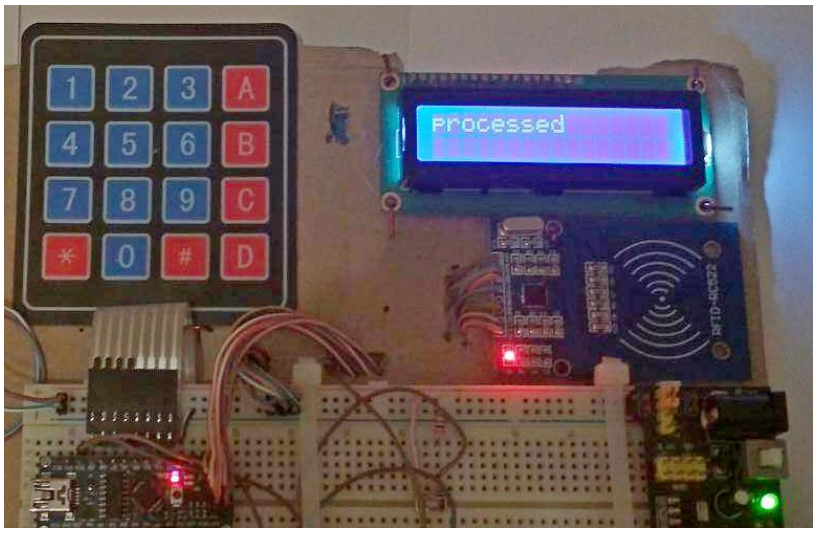

Рисунок 4.30 – Повідомлення про здійснення процесу відвантаження

У разі виникнення позаштатних ситуації треба натиснути кнопку «C» і відвантаження закінчиться достроково. Після завершення на екран виведеться фактичний розмір відвантаження та напис «done» рис. 4.31.

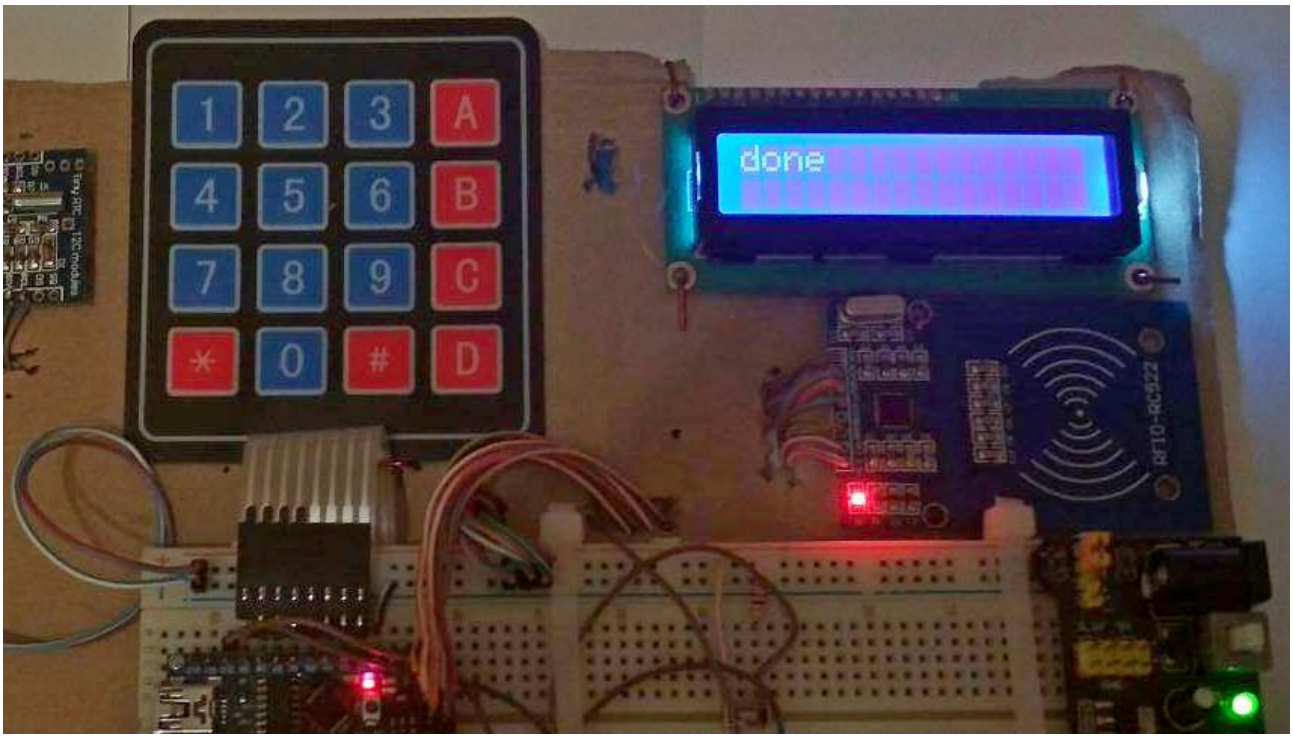

Рисунок 4.31 – Повідомлення про закінчення відвантаження

Окрім основних в функцій в пристрої є сервісне меню в яке можна потрапити через секретну комбінацію клавіш яке дозволяє виконати самодіагностику пристрою, та вивести результат діагностики на екран.

ВИСНОВОК

Під час дипломного проектування були виконані усі поставлені задачі.

Було проведено аналіз предметної області, були виявлені основні проблеми які вирішить дипломний проект, досліджено актуальність проблем.

Проведено дослідження ринку на рахунок вже реалізованих систем і пристроїв, виявлено їх переваги та недоліки.

Сформовано мету і ціль дипломного проекту, розроблено технічне завдання з детальним описом усіх потрібних функцій і технічних особливостей.

Досліджено технології та засоби за допомогою яких можливо виконати дипломний проект, обрано технології які буду використані під час розробки.

Виконано структурно-функціональне моделювання роботи ІС, проаналізовано предметну область та створено модель бази даних, визначено варіанти використання системи.

Реалізовано web додаток для функціонування і адміністрування інформаційної системи, який можуть користуватися і звичайні користувачі і адміністратори системи.

Розроблено пристрій обліку і відвантаження, який працює у парі web додатком.

Виконано усі вимоги технічного завдання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джалота, П. Управление проектами в области информационных технологий / П. Джалота. - М.: Лори, 2014. - 224 с.

2. Гультяев, А.К. MS Project 20010 Professional. Управление проектами: Самоучитель / А.К. Гультяев. - СПб.: Корона Принт, 2009. - 512 с.

3. Коваленко, С.П. Управление проектами: Пректическое пособие / С.П. Коваленко. - Мн.: Тетралит, 2013. – 192 с.

4. В чем отличие сайта от веб-приложения? [Електронний ресурс]. – Точка доступу: URL: http://evriqum.ru/work/web-applications – В чем отличие сайта от веб-приложения?

5. Довідник функцій PHP [Електронний ресурс]. – Точка доступу: URL: https://www.php.net/manual/ru/funcref.php – Довідник функцій PHP.

6. Довідник HTML [Електронний ресурс]. – Точка доступу: URL: http://htmlbook.ru/html – Довідник HTML.

7. Apache HTTP Server [Електронний ресурс]. – Точка доступу: URL: https://ru.wikipedia.org/wiki/Apache_HTTP_Server – Apache HTTP Server.

8. Торгові автомати як стартап [Електронний ресурс]. – Точка доступу: URL:https://www.prostobiz.ua/biznes/biznes_start/stati/torgovye_avtomaty_kak_startap_kak_i_pochem_voyti_v_vendingovyy_biznes – Торговые автоматы как стартап.

9. Вендінг [Електронний ресурс]. – Точка доступу: URL: https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D0%B4%D0%B8%D0%BD%D0%B3 – Вендінг.

10. Торговий автомат власними руками [Електронний ресурс]. – Точка доступу: URL: https://arduinoplus.ru/arduino-vending/ – Торговый автомат своими руками

11. Початок роботи з Bootstrap [Електронний ресурс]. – Точка доступу: URL: https://bootstrap-4.ru/docs/4.3.1/getting-started/introduction/ - Введение, быстрый старт в Bootstrap.

12. Документація OpenServer [Електронний ресурс]. – Точка доступу: URL: https://ospanel.io/docs/ - Руководство пользователя.

13. Визначення MVC методології побудови web додатків [Електронний ресурс]. – Точка доступу: URL: https://ru.wikipedia.org/wiki/Model-View-Controller - Model-View-Controller.

14. Створення ER діаграм [Електронний ресурс]. – Точка доступу: URL: http://inf-teh-lotos.ru/sozdanie-er-diagramm – Создание ER Диаграм.

15. Що таке IMEI [Електронний ресурс]. – Точка доступу: URL: https://uk.wikipedia.org/wiki/IMEI – IMEI.

16. Документація бібліотеки Leaflet [Електронний ресурс]. – Точка доступу: URL: https://leafletjs.com/reference-1.5.0.html - Leaflet API reference.

17. М’якшило О. М., Харкянен О. В. Проектування інформаційних систем : конспект лекцій для студ. освіт. ступ. "Бакалавр" спец. 122 "Комп’ютерні науки" ден. та заоч. форм навч. /; Нац. ун-т харч. технол. - Київ : НУХТ, 2018. - 47 с.

18. Визначення мови програмування JavaScript [Електронний ресурс]. – Точка доступу: URL: https://uk.wikipedia.org/wiki/JavaScript - JavaScript (JS)

19. Козак О.Л. Опорний конспект лекцій з курсу ― Аналіз вимог до програмного забезпечення‖ для студентів напрямку підготовки ―Програмна інженерія‖ / О.Л. Козак. – Тернопіль, 2011. – 56 с.

20. Розділення візуалізаціїї та бізнес логіки [Електронний ресурс]. – Точка доступу: URL - https://bit.ly/31EqiwL - Разделение визуализации и бизнес-логики.

21. Thing-Model-View-Editor [Електронний ресурс]. – Точка доступу: URL: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html - Trygve/MVC.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної система обліку і відвантаження рідких вантажів

Суми 2019

Призначення й мета створення інформаційної системи

Призначення інформаційної системи

Інформаційна система (ІС) призначена автоматизувати процеси відвантаження рідких вантажів, зберігати журнал відвантаження.

Мета створення інформаційної системи –

Позбутися від частини технічного персоналу, забезпечити єдину систему обліку, унеможливити непомічені махінації з вантажем під час його транспортування, на основі статистичних даних здійснювати подальше бізнес планування.

Цільова аудиторія

У цільовій аудиторії ІС можна виділити наступні групи:

- Керівник компанії по роздрібному збуту.
- Водії – оператори автоцистерн.
- Інженери з підтримки і сервісного обслуговування пристроїв обліку.
- Адміністратори системи.
- Інші зацікавлені сторони.

Вимоги до інформаційної системи

Вимоги до структури й функціонування інформаційної системи

ІС повинна бути реалізована у вигляді WEB додатку, який дозволить моніторити запаси і потреби кожного з роздрібних пунктів збуту. WEB додаток повинен мати два основних блоки, перший відповідає за обробку запитів від кінцевих пристроїв обліку, другий повинен давати інструменти для адміністрування системи і проведення аналітичної роботи.

Вимоги до персоналу

Вимоги до персоналу, який здійснює роботу з ІС, визначаються специфікою роботи з персональним комп'ютером. Працівники повинні:

- Володіти принаймні початковими навичками роботи з обчислювальною технікою та периферійними пристроями, з ПК зокрема;
- Знати і дотримуватися правил техніки безпеки при роботі з електротехнічним обладнанням;
- Мати допуск до роботи з комп'ютером за станом здоров'я.

Вимоги до збереженні інформації

Адміністратор ІС повинен мати можливість створення резервних копій бази даних. Бажано, щоб процедура проводилась автоматично з певною періодичністю.

Вимоги до розмежування доступу

Дані ІС є корпоративною таємницею з обмеженим доступом.

Користувачів ІС можна поділити на 4 групи відповідно до прав доступу:

- Пристрій обліку;
- Інженер з технічної підтримки;
- Менеджер;
- Адміністратор.

Пристрій обліку має права:

- Виконувати запит на перевірку облікових даних для доступу до системи;
- Додавати записи про відвантаження вантажу або навпаки поповнення запасів.

Інженер з технічної підтримки має права:

- Видаляти записи в разі виникненні помилок;
- Створювати нові записи у ручному режимі;
- Редагувати квоти за різними обліковими записами.

Менеджер має права:

- Переглядати журнали обліку;
- Переглядати статистичні дані за різними пунктами роздрібного збут;
- Редагувати квоти за різними обліковими записами;
- Формувати звіти.

Адміністратор може виконувати всі вищевказані дії а також:

- Додавати, редагувати та видаляти користувачів ІС;
- Керувати налаштуваннями ІС.

Вимоги до функцій, виконуваних ІС

Основні вимоги

Структура ІС

ІС повинена складатися з наступних розділів:

- Головна – виводиться журнал останніх відвантажень;
- Пошук – розділ пошуку за заданими критеріями.
- Звіт – розділ формування звітів за заданими критеріями.
- Адмін-панель – розділ налаштування ІС та користувачів.

Навігація

Користувацький інтерфейс ІС повинен забезпечувати повне та інтуїтивно зрозуміле представлення відвантажень за всіма складами. Навігаційні елементи повинні забезпечувати однозначне розуміння користувачем їх змісту: посилання на сторінки повинні бути мати заголовок, умовні позначки відповідати загальноприйнятим. Графічні елементи навігації повинні бути мати альтернативний підпис.

ІС повинна забезпечувати навігацію по всіх доступних користувачеві розділах та відображати відповідну інформацію. Для навігації повинна

використовуватися система контент-меню. Меню повинне являти собою текстовий блок (список гіперпосилань) у лівій колонці або у верхній частині сторінки (залежно від затвердженого дизайну).

Відображення та внесення інформації до системи (відвантажень)

Інформаційне наповнення має бути, з одного боку, повним, а з другого - не містити зайвих відомостей, недоречних повторів тощо. Сторінка зі списком останніх відвантажень повинна формуватися програмним шляхом на підставі інформації у базі даних на сервері.

Повинна бути можливість додавання та редагування відвантажень інженером з підтримки ІС за допомогою web-інтерфейсу.

Система навігації (структура ІС)

Взаємозв'язок між розділами системи (структура ІС) представлено на рисунку А.1.

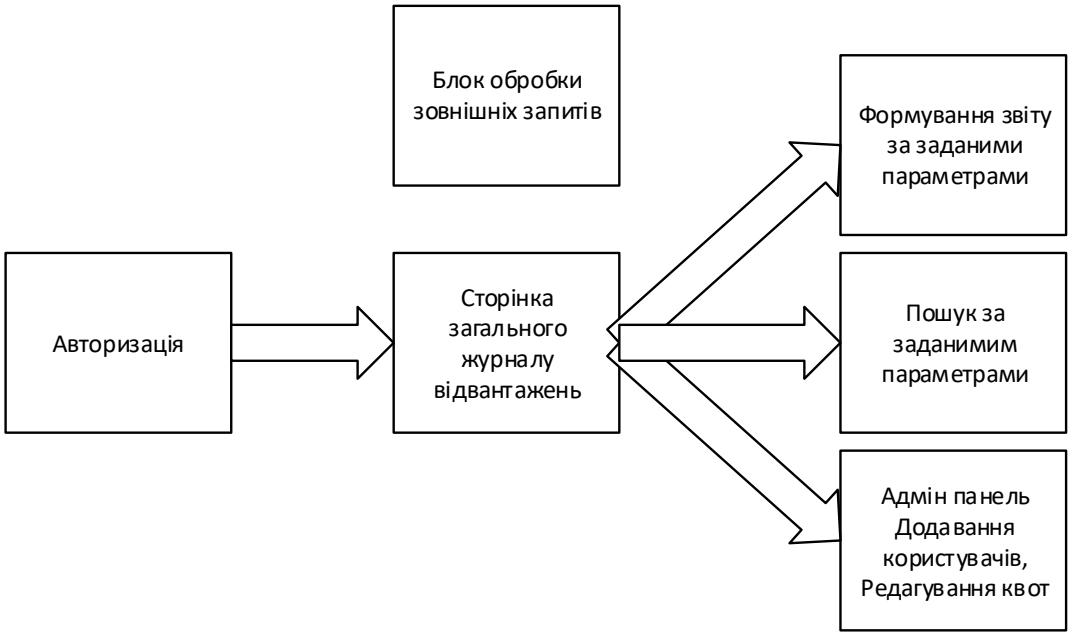

Рисунок - А.1.

Сторінка авторизації містить поля для введення логіну та паролю а також кнопку для входу.

Сторінка загального журналу містить список останніх подій в системі, логотип, кнопку для виходу, навігаційне меню для переходу на інші сторінки, сторінка дає можливість видаляти, додавати, і редагувати записі в журналі.

Сторінка формування звіту будує звіт за заданими параметрами, параметри обираються у випадаючому списку, містить кнопку «Зробити звіт».

Сторінка пошуку, виводить записи з журналу які співпадають з параметрами пошуку.

Адмін панель використовується для адміністрування користувачів системи, дозволяє додавати, видаляти, та редагувати усі дані про кристувачів.

Вимоги до функціональних можливостей

Інформаційна система повинна підтримувати в актуальному стані, коректно працювати при великій завантаженості і давати адміністраторам можливість:

- Авторизуватися у системі з різними правами доступу;
- Давати можливість пристроям обліку створювати записі в журналі;
- Редагувати записи в журналі;
- Надавати зведену інформацію з журналу;
- Виконувати пошук записів в журналі;
- Формувати звіти.
- Створювати резервні копії інформації, що обробляється системою.
- Адмініструвати систему.

Функціональні можливості розділів

Сторінка авторизації дозволяє увійти до ІС користувачеві з його правами доступу. Для успішної авторизації треба ввести логін та пароль надані адміністратором.

Після успішної авторизації користувач потрапляє на головну сторінку. На головній сторінці будуть представлені наступні елементи:

- Кнопка створити новий запис в журнал;
- Кнопка переходу на сторінку пошуку записів;
- Кнопка переходу на сторінку звітів;

- Список записів в журналі;
- Кнопки навігації по списку записів;
- Інформація про авторизованого користувача.
- Кнопка налаштувань ІС

У журналі записів у кожному записі містяться такі дані та кнопки:

- Номер запису;
- Інформація про тип запису додавання чи віднімання;
- Номер електронного ключа за яким отримали доступ до системи;
- Об’єм рідини що відвантажили;
- Дата та час відвантаження;
- Ідентифікатор пристрою обліку;
- Кнопка «Редагувати» запис;
- Кнопка «Зберегти» виконані зміни у записі;
- Кнопка видалення запису;
- Кнопка додавання запису;

При натисканні кнопки «Редагувати» стає можливо вносити зміни до поточного запису. Для збереження змін необхідно натиснути кнопку «Зберегти».

На сторінці пошуку будуть представлені наступні елементи:

- Форма вводу номеру запису;
- Форма вводу номеру пристрою;
- Форма вводу типу відвантаження;
- Кнопка переходу на сторінку звітів;
- Кнопка переходу на сторінку списку записів;
- Список знайдених записів.

На сторінці звітів будуть представлені наступні елементи:

- Форма вводу номеру пристрою;
- Форма для вибору періоду для формування звіту;
- Кнопка переходу на сторінку пошуку записів;
- Кнопка переходу на сторінку списку записів;

- Список попередньо сформованих звітів.

Загальні вимоги

Система не повинна бути перенавантаженою надлишковою інформацією. Текст повинен бути зручним для читання. Кольори та елементи на сайті не повинні відволікати увагу.

Типові навігаційні й інформаційні елементи

- Шапка сайту.
- Головне меню та елементи навігації.
- Основне поле контенту.

Шапка сайту

Шапка сайту повинна містити логотип і назву сайту. Логотип є посиланням на головну сторінку сайту.

Головне меню та елементи навігації.

Головне меню повинне розташовуватися у верхній частині вікна (зправа від шапки) і містити посилання та форми вводу для переходу на всі розділи системи.

Елементи навігації повинні розташовуватися у нижній частині вікна (під областю списків) і містити елементи навігації для переходу між сторінками журналу записів.

Основне поле контенту

Основне поле контенту повинне розташовуватися в центрі сторінки. У цьому полі відображається основний зміст обраного розділу. Стильове оформлення матеріалів і їх елементів (посилань, заголовків, основного тексту, зображень, форм, таблиць і т.п.) повинне бути єдиним для всього веб-сайту.

Вимоги до видів забезпечення

Вимоги до інформаційного забезпечення

Інформаційне забезпечення - це сукупність єдиної системи класифікації і кодування інформації, уніфікованих систем комунікації, схем інформаційних потоків, що циркулюють в організації та методологія побудови баз даних. Його

призначення - це своєчасне формування і видача достовірної інформації для прийняття управлінських рішень.

Для реалізації ІС буде використана реляційна СУБД. Тому повинна бути розроблена логічна структура реляційної бази даних, на основі якої буде здійснюватися рішення задачі.

Вимоги до лінгвістичного забезпечення

Сайт повинен бути виконаний українською мовою. Усі звіти та бланки для друку повинні бути виконані українською мовою.

Вимоги до програмного забезпечення

ІС складається з серверної, клієнтської частини та пристроїв обліку. Для їх правильного функціонування програмне забезпечення (ПЗ) повинно швидко та ефективно обробляти інформаційні потоки системи.

Реалізація серверної частини відбувається з використанням:

- Apache 2.4
- PHP 5.6
- MySQL 5.6

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

– Операційна система Windows 7 або вища.

– Веб браузер версія якого вийшла після 2016 року.

– Включена підтримка javascript і cookies.

Вимоги до апаратного забезпечення

Апаратне забезпечення серверної частини, для початку функціонування системи повинне задовольняти наступним вимогам:

1 Гб вільної RAM.

3 ГБ вільного місця на HDD.

Підтримка демонів (програм) Apache, MySQL.

При підвищенні навантаження на систему, може з’явитися потреба збільшити об’єм апаратних ресурсів.

Апаратне забезпечення клієнтської частини повинне забезпечувати підтримку програмного забезпечення клієнтської частини, зазначеного в пункті про вимоги до програмного забезпечення.

Склад і зміст робіт зі створення сайту

Докладний опис етапів роботи зі створення ІС наведено в табл. А.1.

Таблиця А.1 – Етапи створення ІС

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Розробка бази даних та основних функціональних блоків системи.	10 дні
2	Створення основного оформлення сайту.	2 дні
3	Розроблення та реалізація блоку Авторизації до ІС.	2 дні
4	Розробка функціоналу створення нового запису.	2 дні
5	Реалізація відображення та сортування списку записів.	2 дні
6	Реалізація пошукового функціоналу.	2 дні
7	Аналіз, проектування та реалізація форм звітів.	3 дні
8	Тестування функціоналу веб сайту.	2 день

	Загальна тривалість робіт (з урахуванням строку налагодження і тестування) і строк здачі проекту	28

Вимоги до складу й змісту робіт із введення сайту в експлуатацію

Для створення умов функціонування, при яких гарантується відповідність створюваного сайту вимогам сьогодення ТЗ і можливість його ефективної роботи, в організації Замовника повинен бути проведений певний комплекс заходів.

Для переносу ІС на хостинг необхідно, щоб параметри хостинга відповідали вимогам, зазначеним у ТЗ. На хостинг переноситься програма (сайт), зверстаний шаблон дизайну, структура і наповнення бази даних з подальшим використанням системи.

ДОДАТОК Б

Планування робіт

Деталізація мети методом SMART

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Розробити WEB додаток який дозволить вести журнал обліку рідин на складах, рідин що транспортуються між складами, журнал збуту товару кінцевому користувачу. Розробити апаратну платформу, що буде давати змогу взаємодіяти з терміналом кінцевому користувачу, відправляти дані на сервер та отримувати їх від нього.
Measurable (вимірювана)	Результатом роботи проекту є оцінка замовника, кількість коштів яку, допоможе зекономити система
Achievable (досяжна)	Реалізації системи здійснюється за допомогою технологій HTML, PHP, MySQL, пристрої обліку функціонують на база процесора AVR.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у терміни, що були оговорені замовником проекту. Проект повинен бути виконаний згідно з календарним планом.

Планування змісту робіт

Структурна декомпозиція робіт (Work Breakdown Structure) - називають представлення проекту, виконане у вигляді ієрархічної структури робіт, що досягається за допомогою послідовної декомпозиції. Інструмент спрямований на детальне планування, оцінку вартості, визначення та розподіл персональної відповідальності виконавців. Тобто, на основні робот і результатів, що визначають зміст проекту. WBS структура рисунок Б.1.

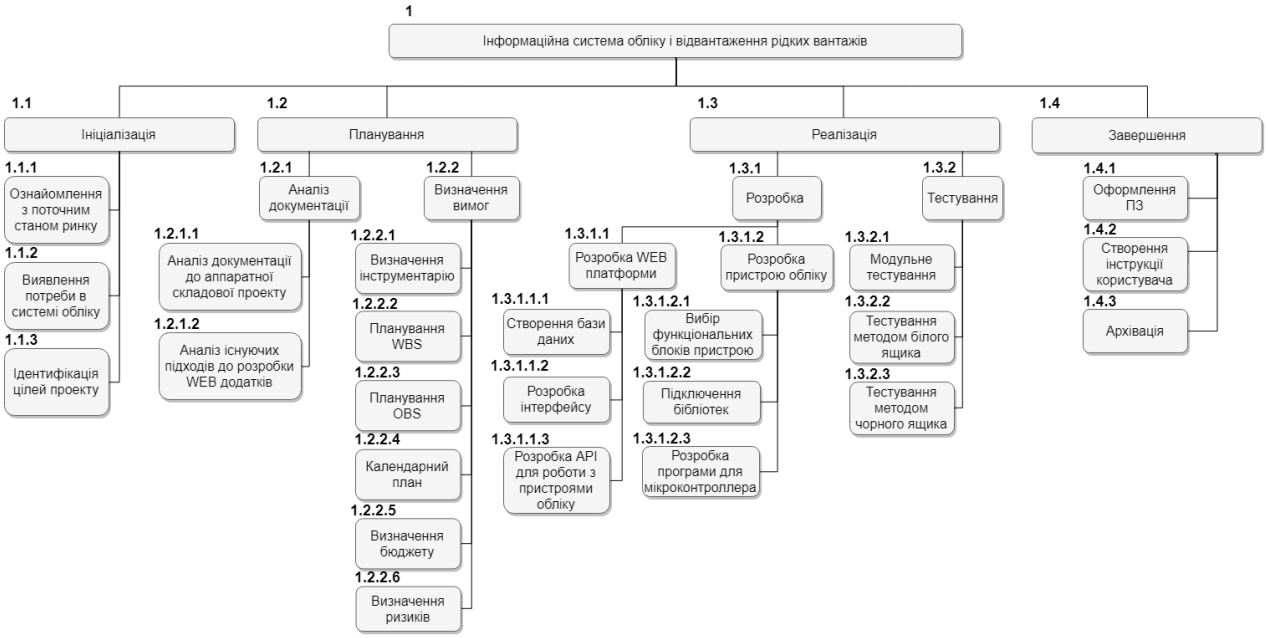

Рисунок Б.1 –WBS структура системи

Планування структури виконавця для впровадження готового проекту

Говорячи про організаційну структуру, ми маємо на увазі концептуальну схему, навколо якої організовується група людей, основу, на якій тримаються всі функції.

Організаційна структура підприємства - це, по суті керівництво для користування, яке пояснює, як організація збудована і як вона працює. Якщо

говорити конкретніше, то організаційна структура описує, як в компанії приймаються рішення і хто є її лідером. OBS структура рисунок Б.2.

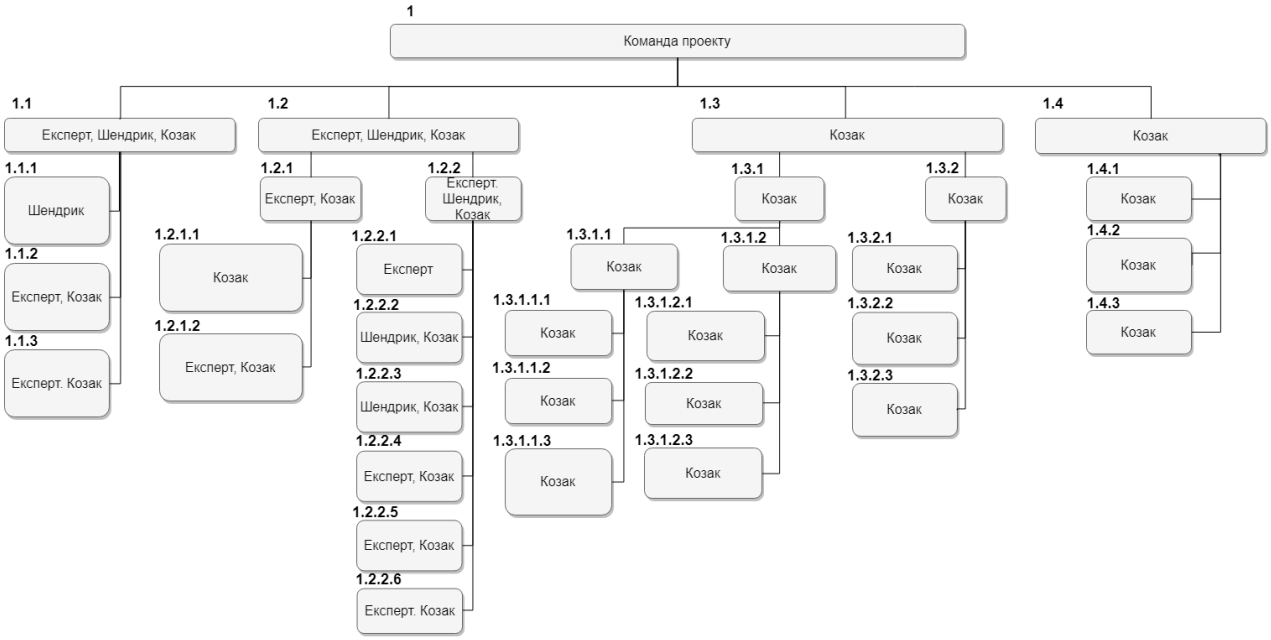

Рисунок Б.2 – OBS структура системи

2.3 Побудова матриці відповідальності

Матриця RACI є простий інструмент, який використовується для визначення значень і обов'язків та уникнення плутанини при виконанні завдань або процесів. Використовується при управління проектами і для показу обов'язків в станах "AS-IS" і "TO-BE".

Матриця відповідальності являє собою особливий метод визначення функціональних областей, ключових напрямків діяльності, критеріїв прийняття управлінських рішень, де існують неясності. Всі розбіжності, що виникають в ході даного процесу, можуть бути винесені на загальне обговорення та згодом вирішені шляхом прийняття колективного рішення.

Беручи за основу OBS та WBS структури була побудована матриця відповідальності проекту, яка реалізована у вигляді таблиці.

Таблиця Б.2 – Матриця відповідальності

WBS\OBS	Козак	Шендрик	Експерт
Ознайомлення з поточним станом ринку			
Виявлення потреби в системі обліку			
Ідентифікація цілей проекту			
Аналіз документації до апаратної складової проекту			
Аналіз існуючих підходів до розробки WEB додатків			
Визначення вимог			
Визначення інструментарію			
Планування WBS			
Планування OBS			
Календарний план			
Визначення бюджету			
Визначення ризиків			
Створення бази даних			
Розробка інтерфейсу			
Розробка API для роботи з пристроями обліку			
Вибір функціональних блоків пристрою			
Підключення бібліотек			
Розробка програми для мікроконтролера			
Модульне тестування			
Тестування методом білого ящика			
Тестування методом чорного ящика			
Інструкція користувача			
Оформлення ПЗ			
Створення інструкції користувача			
Архівація			

Розробка PDM мережі

PDM мережа була побудована за допомогою надбудови програми GanttProject. Ця надбудова має назву Pert діаграма. PERT призначений для дуже масштабних, одноразових, складних проектів. Метод має на увазі наявність невизначеності, даючи можливість розробити робочий графік проекту без точного знання деталей і необхідного часу для всіх його складових. PERT був розроблений головним чином для спрощення планування на папері і складання графіків великих і складних проектів. Метод особливо націлений на аналіз часу,

який потрібен для виконання кожної окремої задачі, а також визначення мінімального необхідного часу для виконання всього проекту. Мережу зображено (рис. Б.3).

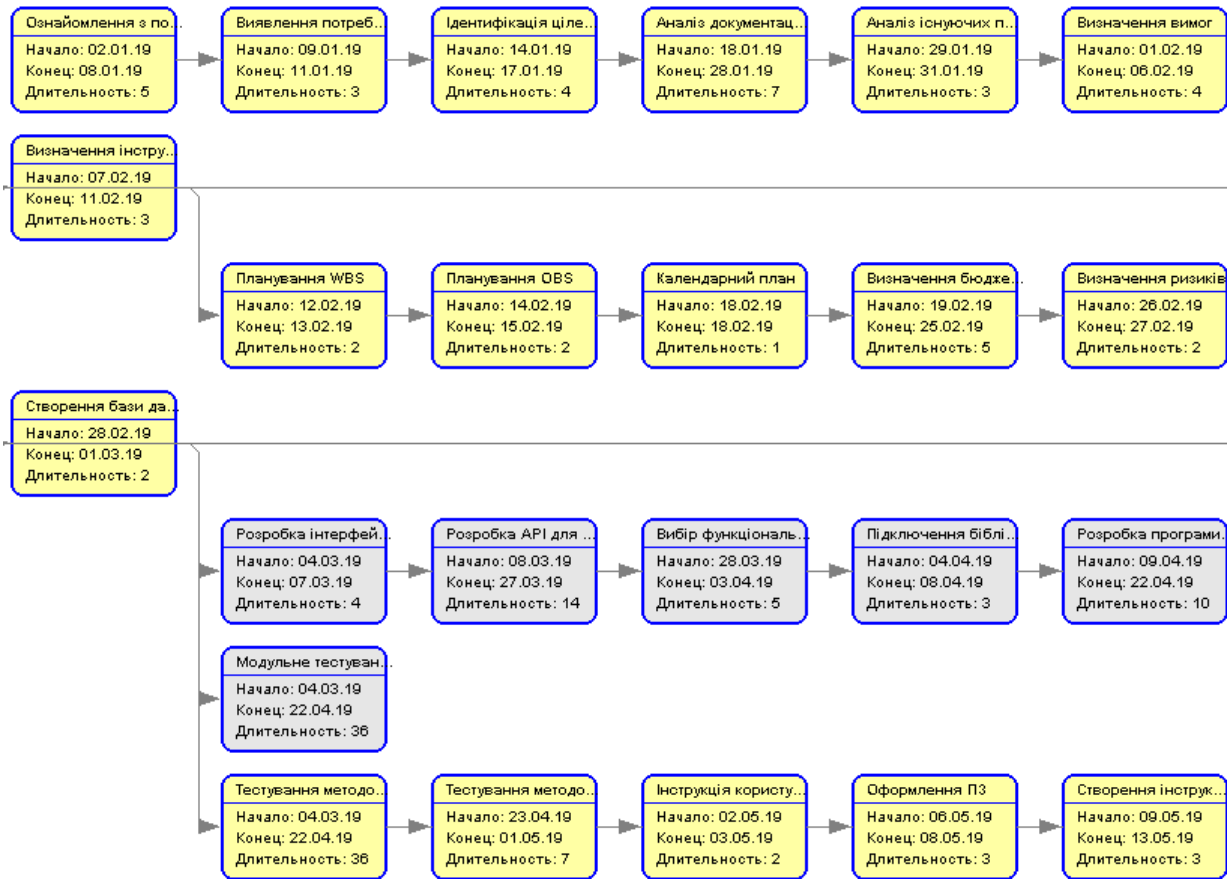

Рисунок Б.3 – PDM-мережа

Побудова календарного графіку виконання ІТ-проекту

Діаграма Ганта - це візуальний спосіб відображення запланованих завдань. Горизонтальні графіки широко використовуються для планування проектів будь-яких розмірів в різних галузях і сферах. Це зручний спосіб показати, яка робота планується до виконання в певний день і час. Дана діаграма також допомагає командам і менеджерам проектів контролювати дати початку і закінчення будь-якого проекту. Все в одному просторі. Діаграму Ганта наведено на рисунку Б.4.

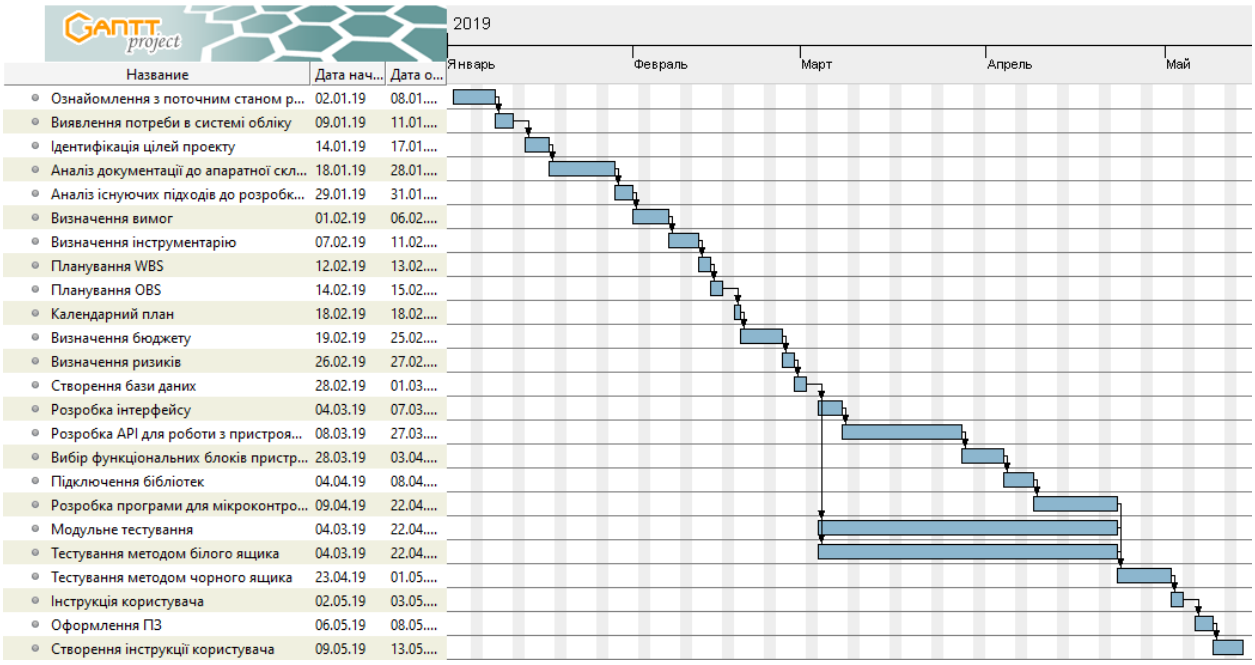

Рисунок Б.4 – Календарний графік робіт

1.1 Управління ризиками проекту

Коли були визначенні всі роботи даного проекту, та спеціалістів котрі будуть розробляти даний продукт, необхідно передбачити всі можливі ризики котрі можуть вплинути на якість та час розробки даного проекту.

Були виділені такі ризики як:

- R1 – зміна ТЗ на етапі розробки;
- R2 – пропущені помилки у ході розробки;
- R3 – недотримання календарного плану;
- R4 – хвороба розробника;
- R5 – некоректне тестування;
- R6 – некоректна робота апаратного забезпечення.

Наступним кроком за допомогою оцінки експертів була визначена ймовірність появи все можливих ризиків, згідно з цим було побудовано таблицю.

Таблиця Б.3 – Ймовірність виникнення ризиків

Ймовірність виникнення	R1	R2	R3	R4	R5	R6
Дуже низька						
Низька						
Середня						
Висока						
Майже достовірна						

Далі було побудовано таблицю можливих втрат при виникненні ризиків.

Потім було зроблено таблицю тих втрат котрі можуть виникнути в процесі проектування та розробки.

Таблиця Б.4 – Втрати при виникненні ризиків

Розмір втрат	R1	R2	R3	R4	R5	R6
Дуже низький						
Низький						
Середній						
Високий						
Дуже високий						

Виходячи з цих двох таблиць, була побудована матриця впливу (ймовірностей і наслідків) ризиків.

В матриці світлим кольором позначено неважливі ризики-білий колір, темнішим – помірні, темним – критичні.

Таблиця Б.5 – Матриця ймовірність-втрати

Ймовірність						
	Дуже висока					
	Висока					
	Середня			R5		R2
	Низька			R3	R6	
	Дуже низька					R1, R4
		Дуже низький	Низький	Середній	Високий	Дуже високий
	Вплив					

Виходячи з цього, було визначено три критичних ризики, такі як:

- R2 – пропущені помилки у ході розробки;
- R1 – зміна ТЗ на етапі розробки;
- R4 – хвороба розробника;
- R6 – некоректна робота апаратного забезпечення.
- R5 – некоректне тестування;

Що до першого ризику, його можна уникнути лиши за допомогою програміста, а саме зміна детальніше проводити модульне тестування, бути уважним при написанні коду. Другий ризик взагалі він малоймовірний, але якщо він все ж таки виникне, то це викличе додаткову роботу і вихід з календарного плану, неможливе без втручання розробника. Третій ризик, це хвороба розробника, малоймовірна але, у разі виникнення може викликати серйозне відставання від графіку. Четвертий ризик некоректна робота апаратного забезпечення, може пригальмувати розробку або у найгіршому випадку знищить частину вже виконаної роботи. П’ятий ризик це не коректне тестування, що може повести за собою певні проблеми з замовником.

ДОДАТОК В

MySQL код створення бази даних.

```
create table meter_devices (
md_id integer not null AUTO_INCREMENT,
md_imei bigint unsigned  not null,
md_value_of_pulse float unsigned not null,
md_date_of_service datetime,
md_installation datetime,
md_seal_number integer unsigned,
md_phone_balance float,
md_phone_number integer unsigned,
md_last_sync datetime,
md_tn_id integer not null,
md_description char(250),
primary key (md_id));

create table tank (
tn_id integer not null AUTO_INCREMENT,
tn_all_capacity integer unsigned not null,
tn_current_fullness integer unsigned not null,
tn_seal_number integer unsigned,
tn_prise_per_cube float(15,2) unsigned,
tn_fs_id integer not null,
tn_tf_id integer not null,
tn_description char(250),
primary key (tn_id));

create table filling_station (
fs_id integer not null AUTO_INCREMENT,
fs_name char(100),
fs_lat float not null,
fs_lon float not null,
fs_description char(250),
```

```
primary key (fs_id));

create table type_fluids (
tf_id integer not null AUTO_INCREMENT,
tf_name char(100) not null,
tf_descriptions char(250),
primary key (tf_id));

create table card_limits (
cl_id integer not null AUTO_INCREMENT,
cl_tf_id integer not null,
cl_cr_number bigint not null,
cl_balance float unsigned not null,
cl_mode enum ('day','week','month','year','static'),
cl_value float unsigned not null,
primary key (cl_id));

create table cards (
cr_number bigint not null,
cr_data blob not null,
cr_cn_id integer not null,
cr_user_name char(50),
cr_user_surname char(50),
primary key (cr_number));

create table consumers (
cn_id integer not null AUTO_INCREMENT,
cn_name char(50),
cn_phone integer unsigned,
cn_contract_number integer unsigned,
cn_status enum ('active','frozen','delete','trust'),
cn_debit float not null,
cn_us_id_admin integer not null,
cn_description char(250),
primary key (cn_id));
```

```
create table users (
us_id integer not null AUTO_INCREMENT,
us_email char(50) not null,
us_password char(50) not null,
us_salt char(50) not null,
us_role enum ('user','admin') not null,
us_status enum ('active','frozen','delete') not null,
primary key (us_id));

create table refill_log (
rl_id integer not null AUTO_INCREMENT,
rl_md_id integer not null,
rl_cr_number bigint not null,
rl_value float unsigned not null,
rl_datetime datetime not null,
primary key (rl_id));

create table change_refill_log (
crl_id integer not null AUTO_INCREMENT,
crl_rl_id integer not null,
crl_rl_md_id integer not null,
crl_rl_cr_number bigint not null,
crl_rl_value float unsigned not null,
crl_rl_datetime datetime not null,
crl_datetime datetime not null,
crl_action_name char(20) not null,
primary key (crl_id));

create unique index ndx_md_id on meter_devices (md_id);
create unique index ndx_tn_id on tank (tn_id);
create unique index ndx_fs_id on filling_station (fs_id);
create unique index ndx_tf_id on type_fluids (tf_id);
create unique index ndx_cl_id on card_limits (cl_id);
create unique index ndx_cr_number on cards (cr_number);
create unique index ndx_cn_id on consumers (cn_id);
create unique index ndx_us_id on users (us_id);
```

```
create unique index ndx_rl_id on refill_log (rl_id);
create unique index ndx_crl_id on change_refill_log (crl_id);

ALTER TABLE meter_devices add foreign key (md_tn_id) references tank (tn_id) on update cascade on delete cascade;
ALTER TABLE tank add foreign key (tn_fs_id) references filling_station (fs_id) on update cascade on delete cascade;
ALTER TABLE tank add foreign key (tn_tf_id) references type_fluids (tf_id) on update cascade on delete cascade;
ALTER TABLE card_limits add foreign key (cl_tf_id) references type_fluids (tf_id) on update cascade on delete cascade;
ALTER TABLE card_limits add foreign key (cl_cr_number) references cards (cr_number) on update cascade on delete cascade;
ALTER TABLE cards add foreign key (cr_cn_id) references consumers (cn_id) on update cascade on delete cascade;
ALTER TABLE refill_log add foreign key (rl_md_id) references meter_devices (md_id) on update cascade on delete cascade;
ALTER TABLE refill_log add foreign key (rl_cr_number) references cards (cr_number) on update cascade on delete cascade;
ALTER TABLE consumers add foreign key (cn_us_id_admin) references users (us_id) on update cascade on delete cascade;
```

ДОДАТОК Г

Блок підключення до бази даних.

```
<?php
class Connect_DB
{
    private static $factory;
    private $db;

    public static function getFactory()
    {
        if (!self::$factory)
            self::$factory = new Connect_DB();
        return self::$factory;
    }

    public function getConnection()
   {
        if (!$this->db)
      {
         try
         {
            $this->db = new PDO("mysql:host=".DB_HOST.";dbname=".DB_NAME,
DB_USER, DB_PASS);
         }
         catch (PDOException $e)
         {
            exit("Проблема при подключении к базе данных!");
         }
      }
        return $this->db;
    }

}

?>
```

Контролер адмінпанелі користувача.

```
<?php

class Controller_Userdb extends Controller
{
    function __construct()
    {
        parent::__construct();
        if(!(User::isAuthorized()))
        {
            header('Location:/login');
    }
        elseif(User::isAdmin())
```

```
        {
            header('Location:/admindb');
        }

    }

    function action_index()
    {
        $this->action_stat();
    }

    function action_stat()
    {
        $data['title']='Панель керування: Статистика';
        $this->view->generate('userdb_stat.php', 'template_userdb.php',$data);
    }

    function action_cards()
    {
        $data['title']='Панель керування: Керування картками';
        $this->view->generate('userdb_cards.php', 'template_userdb.php',$data);
    }
    function action_logbook()
    {
        $data['title']='Панель керування: Журнал відвантажень';
        $this->view->generate('userdb_logbook.php',
'template_userdb.php',$data);
    }

    function action_settings()
    {
        $data['title']='Панель керування: Налаштування';
        $this->view->generate('userdb_settings.php',
'template_userdb.php',$data);
    }

    function action_addconsumer()
    {
        $data['title']='Панель керування: Додати компанію';
        $this->view->generate('userdb_addconsumer.php',
'template_userdbmin.php',$data);
    }

    function action_settingsupdate()
    {
    }
    function action_logout()
    {
        User::logout();
        header('Location:/');
    }

}
```

Контролер адмінпанелі адміністратора.

```
<?php

class Controller_Admindb extends Controller
{

    function __construct()
    {
        parent::__construct();
        if(!(User::isAuthorized()))
        {
            header('Location:/login');
        }
        elseif(!(User::isAdmin()))
        {
            header('Location:/userdb');
        }

    }

   function action_index()
   {
      $this->action_stat();
   }

    function action_stat()
    {
        $data['title']='Панель керування: Статистика';
        $this->view->generate('admindb_stat.php',
'template_admindb.php',$data);
    }

    function action_filling_station()
    {
        $data['title']='Панель керування: Пункти відвантаження';
        $this->view->generate('admindb_filling_station.php',
'template_admindb.php',$data);
    }

    function action_meter_devices()
    {
        $data['title']='Панель керування: Пристрої обліку';
        $this->view->generate('admindb_meter_devices.php',
'template_admindb.php',$data);
    }

    function action_logbook()
    {
        $data['title']='Панель керування: Журнал обліку';
        $this->view->generate('admindb_logbook.php',
'template_admindb.php',$data);
    }
```

```
    function action_consumers()
    {
        $data['title']='Панель керування: Споживачі';
        $this->view->generate('admindb_consumers.php',
'template_admindb.php',$data);
    }

    function action_settings()
    {
        $data['title']='Панель керування: Налаштування';
        $this->view->generate('admindb_settings.php',
'template_admindb.php',$data);
    }

   function action_logout()
   {
        User::logout();
      header('Location:/');
   }

}
```

Клас обробки дій пов’язаних з користувачем.

```
<?php

class User
{
    public $email;
    public $role;
    public $id;
    public $sid;

    function __construct()
    {
        $this->Init();
    }

    static function Init()
    {
      if (!empty($_COOKIE['sid']))
      {
         session_id($_COOKIE['sid']);
      }
        session_start();
    }

    static function isAuthorized()
    {
        if (isset($_SESSION['user_id']))
      {
            return true;
```

```
        }
      else
      {
         return false;
      }
    }

    static function passwordHash($password, $salt = null, $iterations = 10)
    {

        $salt || $salt = uniqid();
        $hash = md5(md5($password . md5(sha1($salt))));

        for ($i = 0; $i < $iterations; ++$i) {
            $hash = md5(md5(sha1($hash)));
        }
        return array('hash' => $hash, 'salt' => $salt);
    }

    public function getSalt($email) {
      $conn = Connect_DB::getFactory()->getConnection();
      $query = "select us_salt from users where us_email = :email limit 1";
        $sth = $conn->prepare($query);
        $sth->execute(array(":email" => $email));
        $row = $sth->fetch();
      $conn=null;
        if (!$row) {
            return false;
        }
        return $row["us_salt"];
    }

    static function authorize($email, $password, $remember=false)
    {
      $conn = Connect_DB::getFactory()->getConnection();
        $query = "select us_id, us_email, us_role from users where us_email =
:email and us_password = :password limit 1";
        $sth = $conn->prepare($query); //сделали подготовленній запрос
        $salt = User::getSalt($email);  //вытягиваем соль

        if (!$salt)
      {
            return false; //соли нет выходим
        }
        $hashes = User::passwordHash($password, $salt); //соль есть хешируем

        $sth->execute(array(":email" => $email, ":password" =>
$hashes['hash'])); //выполняем подготовленный запрос
        $user = $sth->fetch(PDO::FETCH_ASSOC);
      $conn1=null;

        if (!$user)
      {
            return false; //ппользователя нет не авторизовано
```

```
        }
      else
      {
            $_SESSION['user_email']= $user['us_email'];
            $_SESSION['user_id']= $user['us_id'];
            $_SESSION['user_role']= $user['us_role'];
         if($remember) User::saveSession();

        }
        return true;
    }
    static function logout()
    {
        if (isset($_SESSION['user_id'])) {
            unset($_SESSION['user_id']);
            unset($_SESSION['user_role']);
            unset($_SESSION['user_email']);
         session_destroy();
        }
    }
    static function saveSession($http_only = true, $days = 5 )
    {
         $sid =session_id();
            $expire = time() + $days * 24 * 3600;
            $domain = "";
            $secure = false;
            $path = "/";
            setcookie("sid", $sid, $expire, $path, $domain, $secure,
$http_only);
    }
    static function create($email, $password) {
        $user_exists = User::getSalt($email);
        if ($user_exists) {
            return false;
        }
      $conn = Connect_DB::getFactory()->getConnection();
        $query = "insert into users (us_email, us_password, us_salt) values
(:email, :password, :salt)";
        $hashes = User::passwordHash($password);
        $sth = $conn->prepare($query);
        try
      {
            $conn->beginTransaction();
            $result = $sth->execute(array(':email' => $email,':password' =>
$hashes['hash'], ':salt' => $hashes['salt'],));
            $conn->commit();
        } catch (PDOException $e) {
            $conn->rollback();
            echo "Database error: " . $e->getMessage();
            die();
        }
      $conn=null;
        if (!$result) {
            $info = $sth->errorInfo();
```

```
            printf("Database error %d %s", $info[1], $info[2]);
            die();
        }

        return true;
    }
    static function isAdmin()
    {
        if (isset($_SESSION['user_role'])&&($_SESSION['user_role']=='admin'))
        {
            return true;
        }
        else
        {
            return false;
        }
    }

}
```

Клас маршутизації web додатку.

```
<?php
class Route
{
   static function start()
   {
      $controller_name = 'Main';
      $action_name = 'index';
      $routes = explode('/', $_SERVER['REQUEST_URI']);

      if ( !empty($routes[1]) )
      {
         $controller_name = $routes[1];
      }

      if ( !empty($routes[2]) )
      {
         $action_name = $routes[2];
      }
      $model_name = 'Model_'.$controller_name;
      $controller_name = 'Controller_'.$controller_name;
      $action_name = 'action_'.$action_name;

      $model_file = strtolower($model_name).'.php';
      $model_path = "application/models/".$model_file;
      if(file_exists($model_path))
      {
         include "application/models/".$model_file;
      }
      $controller_file = strtolower($controller_name).'.php';
      $controller_path = "application/controllers/".$controller_file;
```

```
        if(file_exists($controller_path))
        {
           include "application/controllers/".$controller_file;
        }
        else
        {
           Route::ErrorPage404();
        }
        $controller = new $controller_name;
        $action = $action_name;

        if(method_exists($controller, $action))
        {
           $controller->$action();
        }
        else
        {
           Route::ErrorPage404();
        }
    }

    function ErrorPage404()
    {
         header('HTTP/1.1 404 Not Found');
        header("Status: 404 Not Found");
        header('Location:/404');
     }

}
```